COMPUTER LANGUAGE™

VOLUME 4, NUMBER 7

JULY 1987

$3.50 Canada $4.50

WILL THE REAL AI LANGUAGE PLEASE STAND UP

PROLOG AND C JOIN FORCES

OBJECT-ORIENTED PROGRAMMING WITH SCOOPS

AI AND THE 80386

WHAT'S HOT AND WHAT'S NOT: PROLOG AND LISP PRODUCTS

PLUS...

AUTOMATING CONVERSION: FORTRAN TO ADA

MULTIWAY BRANCHING IN C

07

0 74470 17345 8

Borland has to be a whole lot better because our competition is a whole lot bigger

At Borland we're competing with giants like Microsoft and Lotus, so we have to try harder. Out-think them, out-smart them, out-pace them, out-perform them—because we obviously can't out-spend them.

What that means to you—which is all that really matters—is that you now have access to the world's fastest compilers, because we were forced to invent them to compete with the giants. We may be smaller than they are, but we're absolutely, positively not slower than they are.

In the long run what really counts is technological excellence.

BI-1126BW

Our technology is so advanced, it's easy to use

With Turbo Pascal® we invented fast compilers, and followed that technological breakthrough with Turbo Prolog®, then Turbo Basic®, then Turbo C.® Building these fast compilers is not an easy task. It was and is "the little guy" taking a giant step. And having transformed Languages with our new superfast compiler technology, we've turned the same power loose onto our Business products like SideKick® and Reflex®:The Database Manager.

"SideKick continues to influence not only the utilities market but the entire software industry."

William Urschel, PC World

When Borland was founded 4 years ago, the software industry technology level was about C– on a scale of A to F. Which was and is a perfect opportunity for a technology-driven company like Borland. Call us "techies," but we're developers, technicians, tinkerers who know how to make programs run faster, do more, be more and let you fully use the hardware power you've paid for.

At Borland, Price is one thing, but Technology is the main thing

There's no sense in the "price being right" if the product is wrong. A useless product is something you can't give away. Technical excellence and superiority always comes ahead of Price. Always has. Always will at Borland. Our technical leadership began when we invented fast compilers and hasn't been matched since.

Turbo Pascal: The world-wide standard

Pascal was asleep before we transformed it with a technical shot in the arm. Our unique ability to create spectacularly fast compilers was the driving force behind Turbo Pascal's worldwide success.

"For the IBM PC, the benchmark Pascal compiler is undoubtedly Borland International's Turbo Pascal."

Garry Ray, PC WEEK

Turbo Basic: BASIC raised to a new power

We've raised BASIC from the dead with our recent high-speed Turbo Basic. Of course, Microsoft will try to sell you their "QuickBASIC," but we think you're interested in "fast," not "quick." Because we're a smaller company, we had to make Turbo Basic the "best BASIC development environment ever written." Otherwise, we'd be out of business. We try harder.

"Borland International's Turbo Basic is unquestionably an outstanding software product."

Giovanni Perrone, PC WEEK

"Turbo Basic is a compiled BASIC. This gives it execution speeds that leave standard interpretive BASICs like BASICA and GW BASIC in the dust."

William Zachmann
COMPUTERWORLD

Turbo Prolog: The natural language of Artificial Intelligence

Turbo-charging Prolog was an enormous challenge. Creating a development environment on an ordinary PC that would rival those found on dedicated AI workstations like Sun and Apollo was deemed impossible. Enter Turbo Prolog. Exit the rest.

"Turbo Prolog has one of the most powerful user interfaces ever seen in a software development system."

Tom Swan
Programmer's Journal

"If you're at all interested in artificial intelligence, databases, expert systems, or new ways of thinking about programming, by all means plunk down your $100 and buy a copy of Turbo Prolog."

Bruce Webster, Byte

Turbo C: Perhaps the most powerful professional development environment ever written

We've brought tomorrow's technology to Turbo C, which is so fast that it makes the rest look like dead Cs. We've given Turbo C a revolutionary user interface making it a wonderful productivity booster. And in keeping with our commitment to open architectures, we are even offering our users the opportunity to license the source code to Turbo C's run-time library.

"Turbo C does look like What We've All Been Waiting For: a full-featured compiler that produces excellent code in an unbelievable hurry . . .
. . . moves into a class all its own among full-featured C compilers . . .
. . . Turbo C is indeed for the serious developer . . . One heck of a buy—at any price."

Michael Abrash
Programmer's Journal

"Borland International's Turbo Pascal, Turbo Basic and Turbo Prolog automatically identify themselves, by virtue of their "Turbo" forenames, as superior language products with a common programming environment. The apellation also means, to many PC users, a "must have" language . . .
. . .To us, Turbo C looks like a coup for Borland."

Garry Ray, PC Week

Reflex: The Database Manager

With Reflex, we brought new eyes and understanding to spreadsheets with unheard-of graphics, charts, plots and analysis. And just as we do in our Language products, we've added our Reflex Workshop™, a "business toolbox" which gives you everything you need to set up and run more than 20 different kinds of business.

"Reflex actually improves on 1-2-3 in several respects."

William Casey, PC Tech Journal

"Reflex and Reflex Workshop may be the best bargain in software today. Highly recommended."

Jerry Pournelle, Byte

Eureka™: The equation solver for Scientists, Engineers, Students and Professionals

Eureka is as much a step forward in equation solving as spreadsheets have been for accounting and planning. Using Eureka with your PC gives you high performance in desktop engineering. A must!

"Eureka is fast, and simple to use."

Michael Miller, InfoWorld

Sprint®: Bringing all that technology together

Word Processors? There are a lot of good ones around. WordPerfect™, WordStar® 4.0, and MicroSoft® Word are good products. But to compete, we had to build a better product. And using some of the technology we pioneered in our fast compilers, we set out to build the best professional word processor ever written.

We spent a lot of time listening to word processor users. Finding out what they liked, and what they really didn't like about the programs they were using.

We looked at all the features of current word processors and figured out how to make them better. That's how we designed Sprint: The Professional Word Processor, which you'll soon discover is faster, easier, smarter than the rest, and probably the most powerful word processor ever written. Sprint is well worth the wait—and you won't be waiting long.

"Sprint: The Word Processor is a many-splendored package."

Ken Greenberg, PC World

Why it's good for you that Borland's in the business of Business

At Borland, small is actually beautiful. Either Joy or Anne or Dyane will answer the phone—but don't get fresh, because they're all happily married. Call Borland and you talk to an actual human being who doesn't give you the big corporate runaround, but the right extension number. You'll get help, whether it's tech support, questions about new products, or whatever. (We've always had a 60-day money-back guarantee, so we don't get too many calls about that.)

When you're small, you try harder, so what we've done is just that.

Borland Software: Technical superiority, innovation and high performance

NEW

Turbo Pascal Numerical Methods Toolbox™

High-performance, state-of-the-art numerical routines for Turbo Pascal® programmers. A must for Scientists, Students and Engineers!

Only $99.95

NEW

Turbo C®

It's the fastest, most powerful, full-featured C compiler at any price. It's the one you've been waiting for!

Only $99.95

NEW

Turbo Prolog Toolbox™

The new collection of efficient, "smart" tools for use with our highly-acclaimed Turbo Prolog,® the natural language of Artificial Intelligence

Only $99.95

NEW

Turbo Basic®

It's the high-speed BASIC you'd expect from Borland. A complete development environment including an amazingly fast compiler, interactive editor and trace debugging system. Compatible with BASICA

Only $99.95

NEW

Eureka: The Solver™

The innovative, easy-to-use equation solver. A must for Scientists, Engineers, Architects, Financial Analysts, Students, Educators and other Professionals

Only $99.95

All Borland products are trademarks or registered trademarks of Borland International, Inc. or Borland/Analytica, Inc. Other brand and product names are trademarks or registered trademarks of their respective holders. Copyright 1987 Borland International BI-1126BW

BORLAND INTERNATIONAL

For the dealer nearest you or to order by phone call

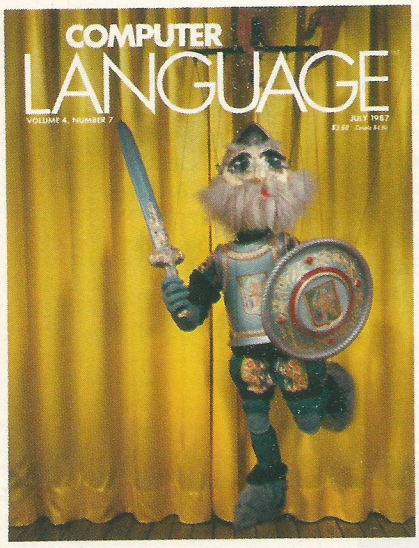
(800) 255-8008

in CA (800) 742-1133
In Canada (800) 237-1136

4585 SCOTTS VALLEY DRIVE SCOTTS VALLEY, CA 95066 (408) 438-8400 TELEX: 172373

CIRCLE 1 ON READER SERVICE CARD

COMPUTER LANGUAGE™



Sir Sampson Knight by Steven M. Overton
Olde World Puppet Theatre ©1982

ARTICLES

PROLOG and C Join Forces — 34
by M. Haytham Matthews
PROLOG and C may come from two different worlds, but here they were combined to take an application created on a LISP machine and reimplement it on a PC/AT. The result? A prototype so strong it went directly into the delivery environment on a trial basis.

Object-Oriented Programming with SCOOPS — 49
by John McGregor
Using this facility of Texas Instruments' PC Scheme, the author develops an implementation of the classic scissors-paper-stone game and illustrates the power of using a procedure to return a procedure.

Will the Real AI Language Please Stand Up? — 58
by Harvey Newquist III
Which programming language is best for writing AI applications: LISP? PROLOG? C? BASIC? FORTRAN? COBOL? It all depends on your point of view.

Automating Conversion: FORTRAN to Ada via Ada — 63
by Lance Eliot and Floyd L. Holliday
The factory approach to software development offers the programmer several advantages through the automation of such mundane tasks as conversion.

Multiway Branching in C — 75
by Alan Filipski
Multiway branches can make your programs both more readable and more efficient. C has two facilities for doing branches: the switch statement and indexed procedure invocation.

PRODUCT WRAP-UP

Death by 1,000 cuts: PROLOG for the PC — 107
LISP PC products: Survival of the species — 137

DEPARTMENTS

Feedback — 9
Programming on Purpose — 17
Who's the boss?
AI-Eye — 27
80386 promises a new age for AI
Public Domain Software Review — 85
Software metrics and EDDY
Designing With Data Bases — 93
Fidelity and the full relational model
Exotic Language of the Month Club — 99
Knowledge representation with CRL
Product BINGO — 105
Advertiser Index — 144

The most brilliant breakthrough in SQL technology since SQL.

XQL is a dramatic step forward in the history of SQL. It's the one unique SQL solution that helps programmers break through to even higher levels of productivity. Powerful yet easy to use, XQL minimizes your coding time and lets you focus on building better applications.

XQL extends the power of Btrieve, SoftCraft's high-performance file manager, by allowing access to multiple records at a time. It frees your application from physical file characteristics by providing true relational capabilities with data independence, data descriptions, data integrity and security.

XQL's three interface levels are a major advance in SQL technology. The first two levels, XQL primitives for maximum efficiency or full SQL statements for maximum convenience, are callable subroutines from BASIC, Pascal and C. The third level lets you enter SQL statements interactively without ever having to write a program.

XQL's extensive DBMS features let you access data by name. Field order is independent of physical location within the Btrieve record. Only records that pass your restrictions are returned—in the sort order you specify. Fields can be computed from other fields or constants. And you can manipulate composite records built from multiple, joined Btrieve files.

XQL offers all the performance and reliability you've come to expect from Btrieve, including LAN support, fault tolerance, comprehensive documentation and expert technical support for trouble-free software development.

Plus, you never pay royalties on your XQL applications.

Put the latest innovation in SQL technology to work for you. Contact SoftCraft.

SoftCraft
A NOVELL COMPANY
P.O. Box 9802, #917
Austin, Texas 78766
(512) 346-8380 Telex 358 200

XQL, $595; Btrieve, $245; multiuser Btrieve, $595. XQL requires Btrieve and PC-DOS or MS-DOS 2.X or 3.X. XQL is a trademark and Btrieve is a registered trademark of SoftCraft, Inc.

CIRCLE 2 ON READER SERVICE CARD

COMPUTER LANGUAGE

EDITOR
Regina Starr Ridley
MANAGING EDITOR
Kathy Kincade
PRODUCT REVIEW EDITOR
Bill Burch
EDITORIAL ASSISTANTS
Judy Barrows, Joanna M. Poole
TECHNICAL EDITOR
John Halamka
CONSULTING EDITOR
Craig LaGrow
CONTRIBUTING EDITORS
Tim Parker, P.J. Plauger, Matthew Rapaport, Don Schuler, Susan Shepard
NATIONAL ADVERTISING DIRECTOR
Ted Bahr
(415-397-1881)
REGIONAL SALES MANAGERS
Mary Christine Gordon
(Midwest—216-493-8733)
Walter Andrzejewski
(East—617-232-5470)
Jeff Eilertsen
(West—415-397-1881)
ADVERTISING SALES ASSISTANTS
Susan McDonald, Linda Pizzotti, Gina M. Dalpra
DIRECTOR OF PRODUCTION
Andrew A. Mickus
OPERATIONS CONSULTANT
Beatrice C. Blatteis
DIRECTOR OF CIRCULATION
Wini D. Ragus
CIRCULATION MANAGER
Donna F. Wilson
CIRCULATION MARKETING MANAGER
Jerry M. Okabe
ADVERTISING TRAFFIC COORDINATOR
Kelly B. Keller
WHOLESALE COORDINATOR
Nicola Sullivan
ART PRODUCTION
Anabella Gonzalez

PUBLISHER
Donald Pazour

CHAIRMAN OF THE BOARD
David R. Stevens
PRESIDENT
Marshall W. Freeman
VICE PRESIDENTS
Charles C. Baake
Leigh M. Freeman
Leonard E. Haas
Barbara M. Hampson
Thomas L. Kemp
Wini D. Ragus
SECRETARY/TREASURER
Miller Freeman III
ASSISTANT TREASURER
Douglas M. Denny
CORPORATE FINANCIAL MANAGER
Charles H. Benz

FOUNDING PUBLISHER
Carl Landau

MILLER FREEMAN publications

Guest Notes

After a protracted childhood in academia, AI is entering a tempestuous adolescence in the commercial marketplace. Like any fortunate child, its parents saw unlimited potential in their offspring; and, like any adolescent venturing out into the world, it has come up against an often less flattering, less indulgent reality.

While the core technologies of AI—symbolic computation, expert systems, and artificial vision, hearing, and speech—undoubtedly have a place in the computing mainstream, finding this place is not as easy as was once assumed.

Prototype systems, nurtured in the protective bosom of their inventors, are still by and large too green for actual deployment in the harsher, more demanding world of business. Unlike the academic computer community, business people don't always want interesting or challenging software. They have a business to run, not a computer.

This fundamental divergence in mindsets is one of the major factors still hampering the penetration of AI technology into corporate America. The commercialization of AI technology is an area where the nascent commercial AI community has a lot to to learn from its "conventional" brethren. Many—perhaps too many—AI companies seem to believe that AI-based solutions are somehow removed from the more mundane, practical issues facing the conventional programming community.

AI technology offers programmers tools to build software solutions to business problems currently unaddressed by more conventional techniques. Despite the aura of glitter and novelty, this technology is intrinsically no more difficult to learn than any other. However, if it is to be successfully integrated into the mainstream, the people currently making the corporate computer world function must be prepared to offer some significant help.

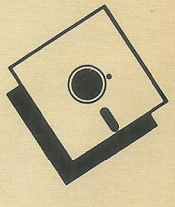
Return on investment is the touchstone by which the businessperson evaluates computer solutions. This doesn't mean simply selling AI-based solutions cheaply. It's more a matter of taking the already well-established base of knowledge about inventing, developing, managing, testing, implementing, and maintaining software in a business environment and applying this knowledge base to the new technology.

Obviously, there may be areas where new rules for evaluating, measuring, and tracking AI-based software projects are needed. Perhaps new methods for justifying the hardware and software purchases required to support these systems must be developed to properly evaluate what value AI-based systems can bring to a corporation.

If these sorts of changes are needed, so be it. They have a much better likelihood of being sold to the corporate decision makers who sign the checks if the people in the driver's seat of commercial data processing—those who have already proven that computers can be a powerful asset to the corporate enterprise—take the lead in understanding and deploying the new technology.

Philip Chapnick
Editor, AI EXPERT

COMPUTER LANGUAGE code can be obtained from CompuServe (type "GO CLM" at the prompt and you'll be in our SIG), USENET ("mod.sources"), and Bulletin Board Services in the U.S. and Canada. Bulletin board parameters are 8 data bits, no parity, 1 stop bit, full duplex, and 300 or 1200 baud.



Complete code listings for all issues since the Premier in 1984 are available from the Chicago BBS and CompuServe. Code on the other BBSs goes back six to 12 months.

(309) 342-5302	Chicago, Ill.	(403) 454-6093	Edmonton, Canada
(517) 355-3276	East Lansing, Mich.	(613) 830-1590	Ontario, Canada
(216) 486-0348	Cleveland, Ohio	(212) 980-0770	New York, N.Y.
(614) 761-1179	Columbus, Ohio	(405) 728-2463	Oklahoma City, Okla.
(206) 848-9232	Seattle, Wash.	(213) 498-6425	Long Beach, Calif.
(415) 487-0310	San Francisco, Calif.		

COMPUTER LANGUAGE *(ISSN 0749-2839) is published monthly by Miller Freeman Publications, 500 Howard St., San Francisco, CA 94105, (415) 397-1881. Please direct advertising and editorial inquiries to this address. Subscription rates for the U.S. are $29.95 for 12 issues, $39.95 for 24 issues, and $49.95 for 36 issues. Canadian/Mexican orders must be accompanied by payment in U.S. funds with additional postage of $6 per year. All other foreign must be prepaid in U.S. funds with additional postage at $15 per year for surface mail and $40 per year for air mail. All subscription orders, inquiries, and address changes should be sent to P.O. Box 11333, Des Moines, IA 50340-1333. For quickest service, telephone toll-free 1-800-341-SERV (in Iowa 1-800-233-IOWA). Second class postage is paid at San Francisco, CA and at additional mailing offices. Please allow six weeks for change of address to take effect.* COMPUTER LANGUAGE *is a registered trademark owned by the magazine's parent company, Miller Freeman Publications.* *All material published in* COMPUTER LANGUAGE *is copyrighted © 1987 by Miller Freeman Publications Inc. All rights reserved. Reproduction of material appearing in* COMPUTER LANGUAGE *is forbidden without written permission.*

More Magic from Blaise.

Turbo C TOOLS™

Magic is easy with Turbo C TOOLS in your bag of tricks. New Turbo C TOOLS™ from Blaise Computing is a library of compiled C functions that allows you full control over the computer, the video environment, and the file system, and gives you the jump on building programs with Borland's new C compiler. Now you can concentrate on the creative parts of your programs.

The library comes with well-documented source code so that you can study, emulate, or adapt it to your specific needs. Blaise Computing's attention to detail, like the use of function prototyping, cleanly organized header files, and a comprehensive, fully-indexed manual, makes Turbo C TOOLS the choice for experienced developers as well as newcomers to C.

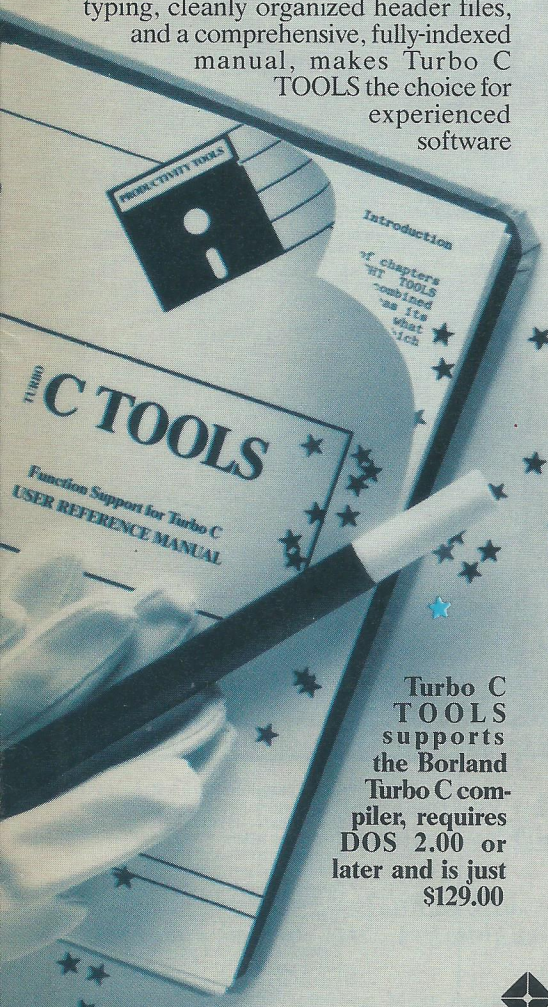
Turbo C TOOLS provides the sophisticated, bullet-proof capabilities needed in today's programming environment, including removable windows, "sidekickable" applications, and general interrupt service routines written in C.

The functions contained in Turbo C TOOLS are carefully crafted to supplement Turbo C, exploiting its strengths without duplicating its library functions. As a result you'll get functions written predominantly in C, that isolate hardware independence, and are small and easy to use.

Turbo C TOOLS embodies the full spectrum of general purpose utility functions that are critical to today's applications. Some of the features in Turbo C TOOLS are:

- **WINDOWS** that are stackable and removable, that have optional borders and a cursor memory, and that can accept user input.
- **INTERRUPT SERVICE ROUTINE** support for truly flexible, robust and polite applications. We show you how to capture DOS critical errors and keystrokes.
- **INTERVENTION CODE** lets you develop memory resident applications that can take full advantage of DOS capabilities. With simple function calls, you can schedule a Turbo C function to execute either when a "hot key" is pressed or at a specified time.
- **RESIDENT SOFTWARE SUPPORT** lets you create, detect, and remove resident utilities that you write with Turbo C TOOLS.
- **FAST DIRECT VIDEO ACCESS** for efficiency, and support for all monitors including EGA 43-line mode.
- **DIRECTORY AND FILE HANDLING** support let you take advantage of the DOS file structure, including volume labels and directory structure.

In addition to Turbo C TOOLS, Blaise Computing Inc. has a full line of support products for Microsoft, Lattice and Datalight C, Microsoft Pascal and Turbo Pascal. Call today for details, and make magic!



Turbo C TOOLS supports the Borland Turbo C compiler, requires DOS 2.00 or later and is just $129.00

THE BLAISE MENU

Turbo POWER TOOLS PLUS *$99.95*
Screen and window management including EGA support; DOS memory control; ISRs; scheduled intervention code; and much more. For Turbo Pascal.

Turbo POWER SCREEN
COMING SOON! General screen management; paint screens; block mode data entry or field-by-field control with instant screen access. For Turbo Pascal.

Turbo ASYNCH PLUS *$99.95*
Interrupt driven support for the COM ports. I/O buffers up to 64K; XON/XOFF; up to 9600 baud; modem and XMODEM control. For Turbo Pascal.

PASCAL TOOLS/TOOLS 2 *$175.00*
Expanded string and screen handling; graphics routines; memory management; general program control; DOS file support and more. For MS-Pascal.

C TOOLS PLUS *$175.00*
Windows; ISRs; screen handling; multiple monitors; EGA 43-line text mode; direct screen access; DOS file handling and more. For MS and Lattice C version 3.00 and later.

LIGHT TOOLS *$99.95*
Windows; ISRs; EGA 43-line text mode; direct screen access; DOS file handling and more. For the Datalight C compiler.

ASYNCH MANAGER *$175.00*
Full featured interrupt driven support for the COM ports. I/O buffers up to 64K; XON/XOFF; up to 9600 baud; modem control and XMODEM. For C or MS-Pascal.

VIEW MANAGER *$275.00*
General screen control; paint screens; block mode data entry or field-by-field control with instant screen access. For C or MS-Pascal.

RUNOFF *$49.95*
Text formatter for all programmers; flexible printer control; user-defined variables; index generation; general macro facility. Crafted in Turbo Pascal.

EXEC *$95.00*
NEW VERSION! Program chaining executive. Chain one program from another in different languages; specify common data areas; less than 2K of overhead.

TO ORDER CALL TOLL FREE
800-333-8087
TELEX NUMBER-338139

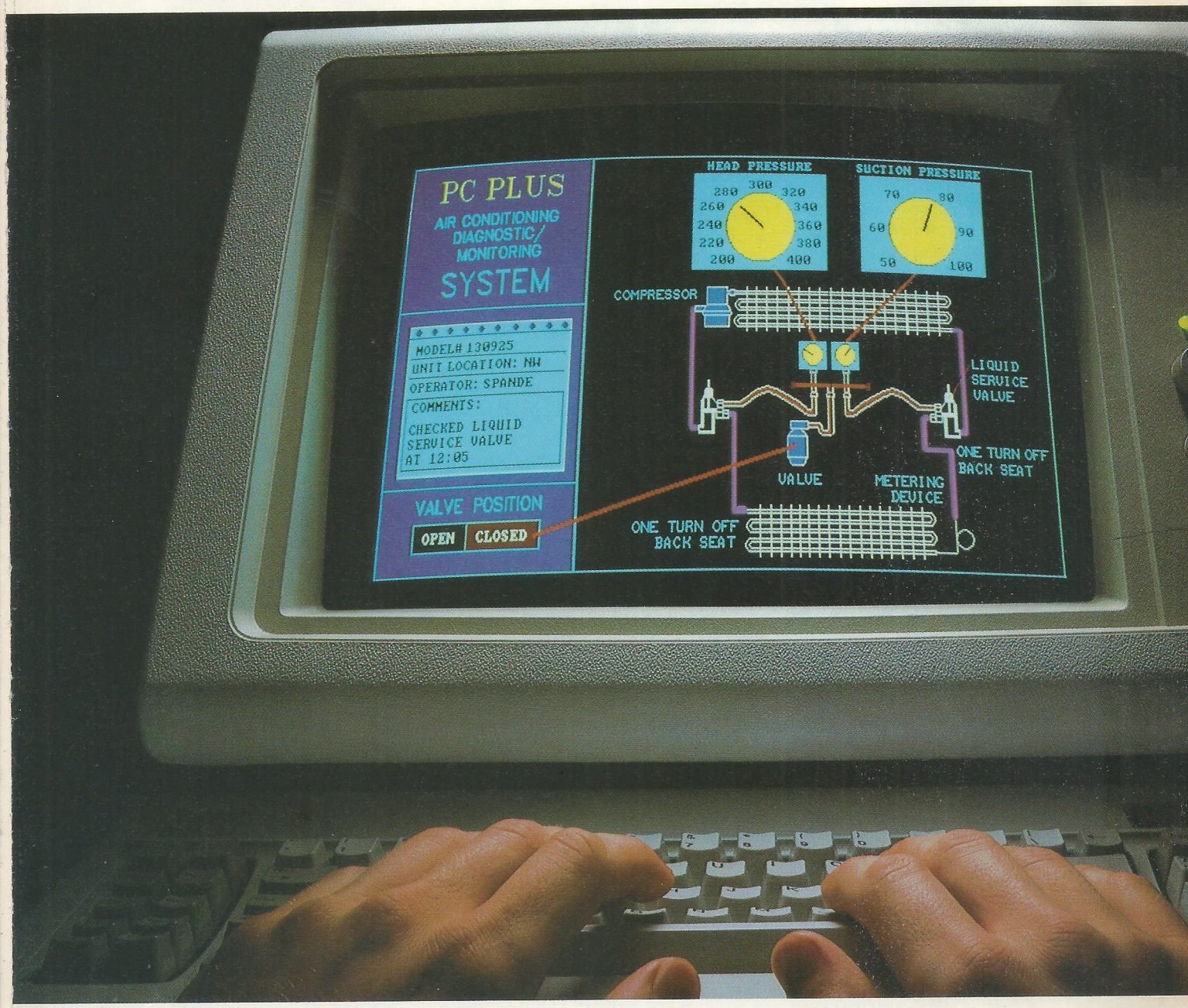
YES! I want to make magic!
Enclosed is $_____ for _____ copies of _____.
☐ Please send me more information on your products.
CA residents add Sales Tax. Domestic orders add $4.00 for UPS shipping, $10.00 for Federal Express standard air.
Name: _____ Phone: (_____) _____
Address: _____
City: _____ State: _____ Zip: _____
VISA or MC#: _____ Exp. Date: _____

Turbo C is a trademark of Borland International.

BLAISE COMPUTING INC.
2560 Ninth Street, Suite 316 Berkeley, CA 94710 (415) 540-5441

CIRCLE 3 ON READER SERVICE CARD

Texas Instruments has system developers need.



"Personal Consultant™ Plus…offers a very fine expert system development and delivery tool that already has a proven record with end-users."

— *Susan Shepard, AI Expert*

Personal Consultant Plus 3.0 Standard Features
- Frames, rules, meta rules and procedures
- Forward/backward chaining
- Confidence factors
- Regression testing and rule tracing
- End-user explanation facilities
- Graphics image capture and display
- Interfaces to dBase™, Lotus 1-2-3™, DOS files, .EXE or .COM programs, "C"
- Complete LISP development environment
- 2-megabyte expanded/extended memory support
- Mouse support
- Context sensitive help
- "Getting Started" tutorial-style manual

Personal Consultant Images
- Optional add-on package to PC Plus (3.0)
- Allows integration of "active images" into

what serious expert Power tools.

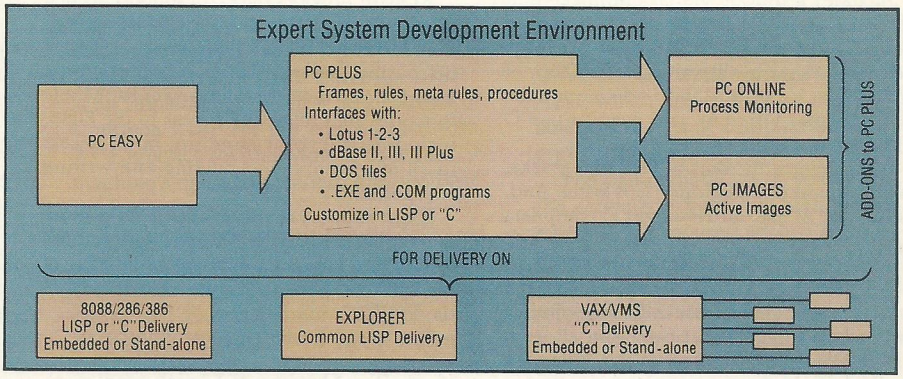
knowledge bases
- Interactive dials, gauges, forms and selection images
- Multiple images can be combined on same screen
- "Getting Started" tutorial-style manual

Personal Consultant Online
- Optional add-on package for PC Plus (3.0)
- ONLINE expert systems that interact directly with process data
- Multiple interfaces to data acquisition and analysis programs
- Knowledge base synchronization with process data
- Functions for historical and predicted trends
- Special user interface/reporting capabilities
- "Getting Started" tutorial-style manual

Among all the expert system development tools available for personal computers today, none deliver the power and flexibility of TI's Personal Consultant series.

Personal Consultant Easy is ideal for getting started, and is upwardly compatible with the higher functionality of PC Plus. For experienced developers, Personal Consultant Plus and its optional add-on enhancements, Online and Images, were designed to help solve a broader range of complex problems.

Application delivery as flexible as the tools themselves.
Delivery can be in LISP for flexibility, or "C"* for maximum speed and portability. Our "C" options support either stand-alone or "embedded" knowledge bases. Options are available for DOS-based PCs, TI's Explorer, and DEC's VAX™ line of multi-user minis running under VMS™.



Personal Consultant Plus. Full power for an affordable price.
At $2,950, PC Plus has proven to be one of the richest and most flexible problem-solving tools available for the development of complex knowledge-based systems. Designed to take advantage of today's more powerful 286/386 DOS-based computers, or TI's Explorer™ Symbolic Processing System, the new 3.0 version of PC Plus provides powerful standard features and a continuing growth path with the addition of either PC Images or PC Online, or both.

Personal Consultant Images. Picture an expert system with interactive graphics.
At $495, PC Images enables developers to create knowledge-based applications that incorporate complex graphical "active images." User-interactive dials, gauges, forms and selection images provide a more exciting visual data input and output style.

Personal Consultant Online. The expert system as part of the process.
At $995, PC Online allows the developer to design expert systems which interact directly with process data, as opposed to input from a human operator. Designed for intelligent process monitoring applications, this optional package helps deliver expertise that is "online all the time."

"Texas Instruments has done more than any other company to educate people about AI, to popularize it, and to make useful AI tools available at reasonable prices."
— Jim Seymour, PC Magazine.

Technical support, training courses and Knowledge Engineering Services are available for the Personal Consultant products. If you have a question about any of our expert system power tools, we have the answer.

Pick up the phone and gain a powerful advantage.
Call **1-800-527-3500** for technical overviews of our products and a PC Plus case histories brochure which details how our power tools are being put to work today.

36106
© 1987 TI
Personal Consultant and Explorer are trademarks of Texas Instruments Incorporated.
dBase is a trademark of Ashton-Tate.
Lotus 1-2-3 is a trademark of Lotus Development Corp.
VAX and VMS are trademarks of Digital Equipment Corporation.
*Available 4Q 1987.

TEXAS INSTRUMENTS

Microsoft Avoids Challenge

We challenged Microsoft to a C compiler duel-to-the-finish, comparing compile, link and execution times, and we offered to stop advertising for two months if they won...

by Roy Sherrill, President, Datalight

Microsoft purchased our C-compiler during February 1987 and we still haven't heard from them. OK, Microsoft, we are extending our challenge deadline from April 1, 1987 to May 15, 1987. After all, the Microsoft ad claims "the fastest C you've ever seen." Your reply, Microsoft!

Walter says Optimum-C is better

Walter Bright, the developer of Optimum C, says that Optimum C would win 7 out of 10 benchmarks as compared to Microsoft C, V.4.0. Walter explained to me that Optimum C includes a unique global optimizer that helps create compact code while increasing execution speed up to 30%. By the way, Borland, Walter is still waiting for his copy of Turbo C,® V.1.0. Borland's ad claims "the fastest, most efficient and easy-to-use C compiler at any price."

After reviewing Borland's benchmarks, Walter claims that Optimum C is faster. And, as for ease of use, all Datalight C compilers have been shipped with a free Learn C program for the last six months. Also, our new EZ Interactive Editor will show you each syntax error in your source code, then compile or "make" and run your program, all from within the editor. OK, so let the Microsoft challenge begin...

We only ask the following...

The benchmark suite will consist of the set of programs that Microsoft supplied to *Computer Language* for their February 1987 C compiler review issue. Microsoft will make available the programs to Datalight at least two weeks prior to the benchmarking. The benchmarking will be between Microsoft C 4.0 and Optimum-C. It will occur at a mutually agreed upon time and place. Interested individuals will be allowed to attend. The benchmarks will be compiled and run on a standard IBM PC-AT.

There will be two separate tests for each program: compile and link speed, and execution speed. For each test, a representative from each company will set up the compiler so that it performs at its best.

The benchmarks will be adjusted so that they take sufficiently long to run, that the tolerance involved in timing them is insignificant. The winner is determined by the compiler with the faster execution times for the majority of the benchmarks. We'd like an answer from Microsoft no later than May 15, 1987.

So what's a global optimizer?

A global optimizer looks at an entire function at once, analyzing and optimizing the whole function. A technique called data flow analysis is used by Optimum-C to gather information about each function. This enables your compute-bound programs to execute as much as 30% faster after global optimization. But, there is one catch...because the global optimizer ruthlessly searches for ways to speed-up execution speed and minimize memory usage, it has relatively slow compile times. No need to worry, though, because you can merely turn the global optimizer off. In fact, you can select all, none, or part of the following optimizations: constant propagation, copy propagation, dead assignment elimination, dead variable elimination, dead code elimination, do register optimizations, global common subexpression elimination, loop invariant removal, loop induction variables, optimize for space, optimize for time, and very busy expressions.

ROM-it Speeds ROM Development

ROM-it provides the extra support needed to speed completion of your ROM applications. ROM-it includes a ROMable C start up routine, and the BLAZE Intel hex utility. ROM-it also includes a library of ROMable functions that allow full access to the 8086, including interrupt handling in C, access to all of memory, and reading and writing I/O ports.

Extra support is also available for remote debugging and multi tasking. The remote debugger supports full symbolic debugging of the application in the target machine, from the PC.

Please call for more information!

Try Optimum-C risk free

Try Optimum-C for 30 days and if you are not 100% satisfied return it for a full refund. *Also available is a C tutorial which is a combination workbook and floppy disk to help lead you through the C language with tutorials, quizzes, and program exercises.

O.K. Microsoft, it's up to you. We've put two months of advertising on the line that says you can't beat Optimum-C to a real test. Your answer, please?

PRICES

Developer's Kit w/ C Tutorial	$ 99
Optimum-C w/ C Tutorial (both with library source)	$139

Add $7 for shipping in US/$20 outside US
COD (add $2.50)

Not Copy Protected

ORDER TOLL-FREE TODAY!

1-800-221-6630

ATTENTION OEMs!

Contact us regarding arrangements.

*Limited offer available exclusively to readers who purchase directly from Datalight.

Microsoft and MS-DOS are registered trademarks of the Microsoft Corporation. Turbo C is a registered trademark of Borland International.

CIRCLE 4 ON READER SERVICE CARD

Magazine Reviewers Shocked by DATALIGHT's Performance...

"Reviewing this compiler was quite a surprise for us. For such a low price, we were expecting a "lightweight" compiler. What we got was a package that is as good as or better than most of the "heavyweights." Datalight C implements a complete C language. It also compiles quickly, doesn't take up much disk space, and looks impressive in the benchmarks."

DR. DOBBS, August 1986

"This is a sharp compiler!... what is impressive is that Datalight not only stole the compile time show completely, but had the fastest Fibonacci executable time and had excellent object file sizes to boot!"

COMPUTER LANGUAGE, February 1986

Optimum-C Version 3.0

NEW!
EZ Interactive Development Environment

NEW!
Inline 8087/80287 Math Support

- Full UNIX System 5 C language plus ANSI extensions
- Fast/tight code via powerful optimizations including common sub-expression elimination
- DLC one-step compile/link program
- Multiple memory model support
- UNIX compatible library with PC functions
- Compatible with DOS linker and assembler
- Third-party library support
- Automatic generation of .COM files
- Supports DOS pathnames, wild cards, and Input/Output redirection
- Compatible with Lattice C version 3.x
- Interrupt handling in C
- Debugger support
- ROMable code support/start-up source

MS-DOS® Support Features

- Mouse support
- Sound support
- Fast screen I/O
- Interrupt handler

MAKE Maintenance Utility

- Macro definition support
- MS-DOS internal commands
- Inference rule support
- TOUCH date manager

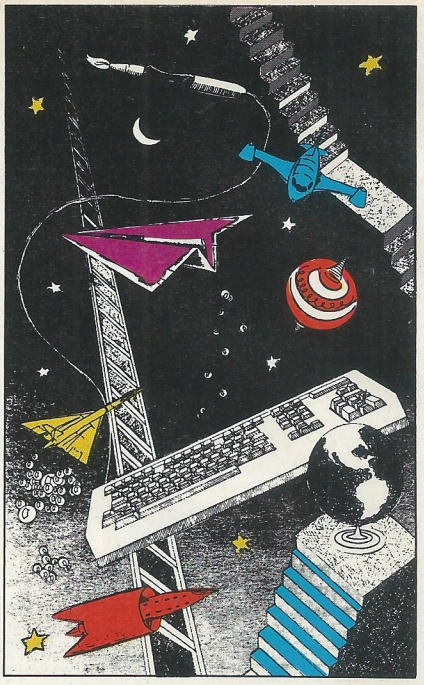
Tools in Source Code

- cat—UNIX style "type"
- diff—Text file differences
- fgrep—fast text search
- pr—Page printer
- pwd—Print working directory
- wc—Word count

Datalight

17505-68th Avenue NE, Suite 304
Bothell, Washington 98011 USA
(206) 367-1803

FEEDBACK



Artwork: Anne Doering

Z8000 defender

Dear Editor:

Having fought my own battles with Intel's militant segmentism, P.J. Plauger's "You must be joking" (Programming on Purpose, *COMPUTER LANGUAGE*, April 1987, pp. 15-19) temporarily changed a grimace into a smile. But I have to point out that it's not fair to find the Z8000 guilty of all of the problems that plague the Intel architectures simply because it too uses segmentation.

At this point in history we are all aware that the linear address space is an awful lot easier on programmers. However, Zilog made the Z8000's segments relatively benign. This microprocessor has 14 general-purpose 16-bit registers. It does not have specific segment registers. Instead, to use it in segmented mode, the general-purpose registers are paired, one providing an offset and the other the segment number, which can be interpreted as a virtual or real address by system hardware.

It is true that the Z8000 has some powerful auto-incrementing and decrementing instructions that will not function properly across a segment boundary, as we would like for accessing, say, an array larger than 64K. But, unlike the 8086 segments, the Z8000's segment register can be the destination of any arithmetic or logic operation, making an explicit carry from incrementing the offset a simple and painless matter. In particular, there is no need to dump another register just to have a scratchpad for address calculations.

It should be pointed out that the Z8000 still has seven truly general-purpose long registers even if all are doubled up for addressing. In contrast, the 8086 really has one general-purpose and six rather limited registers (not counting the stack pointers of either machine). We should also note that each of the Z8000's long registers has a completely independent segment component.

I think that the Z8000's limited market success should probably not be attributed to its use of segmentation but rather to other factors. Still, I want to thank Plauger for lambasting an architecture that has caused more programmers to tear out their hair than perhaps any other.

David McCracken
Aptos, Calif.

Software tools modified

Dear Editor:

P.J. Plauger named the following skills of a software engineer in "Which tool is next?"(Programming on Purpose, *COMPUTER LANGUAGE*, Feb. 1987, pp.15-16):

- Clear understanding of the software cycle: no code before its time.
- Know an accurate specification: don't start until customer signs off.
- Clearly specified deliverables: you must know when you are done.
- Know what adequate documentation is.
- Understand adequate testing.
- And, by the way, know how to design and code programs.

Software engineers, however, are beginning to grudgingly admit the obvious:

- "Plan to throw one away; you will, anyhow." (From *Mythical Man Month).*
- The design always requires changing the specification and the final solution is often found only after you have hacked up a failure. (See "On the Inevitable Intertwining of Specification and Implemetation" *Software Specification Techniques*).
- The customer will always have the last word when he or she sees the system run.

A good software engineer should, therefore:

- Avoid coding like the plague. It takes time and that is the software engineer's most precious commodity. Have your own software tools available or beg, borrow, or buy someone else's.
- Use quality tools to hack up a quick prototype. This is the throwaway version that is built in days or weeks instead of months or years.
- Modify the prototype until the customer is happy. Notice the stuff the customer likes to change and make sure it is client controllable in the delivered product.
- Use performance analysis tools to evaluate throughput and fix or plan to fix bottlenecks. Even with today's hardware, you can't sell a pig.
- The prototype demonstrates the architectural concept, performance characteristics, projects that must be initiated for the final product and yields pretty accurate estimates of the time required to deliver. Now you just follow your nose.

While all this may look like a hacked up version of a software engineer that should probably be thrown away, I submit that the real world is very messy.

Jeffrey V. Sutherland, Ph.D.
Vice President, Product Development
The Saddlebrook Corp.
Cambridge, Mass.

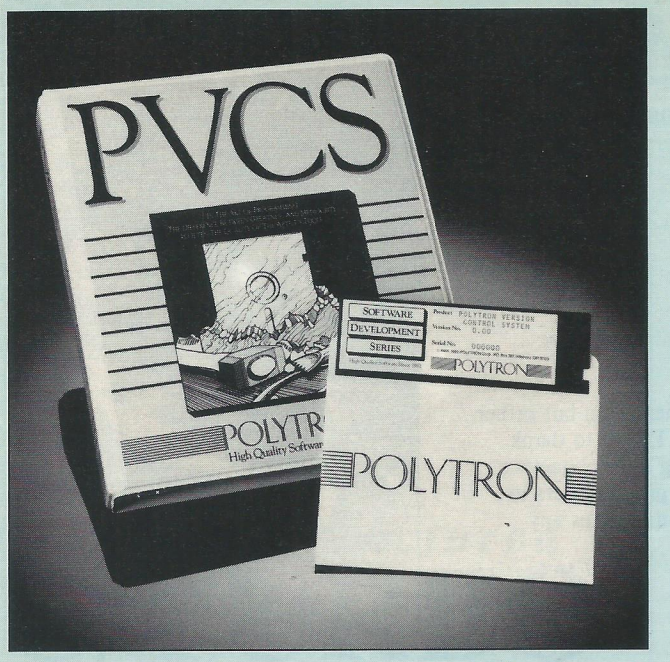
Stallman's response

Dear Editor:

John Heselton's letter "Brilliant but naive"(Feedback, *COMPUTER LANGUAGE*, May 1987, pp. 9-15) shows how a person starting with a basic antipathy to the idea of sharing can misperceive any dissent as the most ludicrous possible strawman. An example of this view is a slave, someone who is forced to do specific jobs by a master. I've never said anyone should be forced to be a programmer or that programmers should never be paid. I am against certain ways of raising the money to pay them, but not all ways.

Heselton's comparison of free software with free gasoline is a good one. Expecting free gasoline is silly because gasoline is used up as you drive and cannot be trivially copied. If software were like that, free software would be silly too. By the way, the "free" in free software refers to freedom, not zero price.

Richard Stallman
Cambridge, Mass.



PVCS

THE MOST POWERFUL AND FLEXIBLE SOURCE CODE REVISION & VERSION CONTROL SYSTEM FOR PCS AND LOCAL AREA NETWORKS

PROGRAMMERS, PROGRAMMING TEAMS, PROJECT MANAGERS, LIBRARIANS AND SYSTEM ADMINISTRATORS FIND PVCS INDISPENSABLE

When large and complex software programs and systems are being developed on personal computers, effective management of the revisions and versions becomes critical. The POLYTRON Version Control System (PVCS) simplifies this process and lets you effectively control the proliferation of code changes. We used UNIX SCCS and RCS as models. However, our own experience, and the input of hundreds of PC programmers has enabled us to significantly improve upon these models.

PVCS provides many powerful functions including:

* Storage & Retrieval of multiple revisions of text.
* Maintenance of a complete history of changes.
* Maintenance of separate lines of development using branching.
* Resolution of access conflicts.
* Merging of simultaneous changes.
* Modules can be retrieved by their own revision number, system version number, or specified date.
* Efficient disk storage. PVCS uses a very intelligent difference detection technique that minimizes the amount of disk space required to store a new version.

PVCS Maintains System Integrity

PVCS prevents system corruption that could ordinarily result from security breaks, user carelessness or malfunctions. The levels of security can be tailored to meet the needs of your project.

The Preferred Version Control System

PVCS is now being used in personal computer software development environments at hundreds of high-technology corporations including:

Hewlett-Packard	3 Com
GTE Data Services	IBM
Sperry	Texas Instruments
Westinghouse Electronics	ROLM Corp.
Nestar Systems	The Aerospace Corp.

Maintains Source Code Written In ANY Language.
Requires DOS 2.0 or higher. Compatible with the IBM PC, XT, AT and other MS-DOS PCs.

Only PVCS meets the needs of Independent Programmers and Corporations. Once you standardize on PVCS, the "Logfiles" used to track and monitor changes are interchangable between any PVCS product. You will receive full credit for your initial purchase if you upgrade to a higher-priced PVCS.

Personal PVCS — Offers most of the power and flexibility of the Corporate PVCS, but excludes the features necessary for multiple-programmer projects. **$149**

Corporate PVCS — Offers additional features to maintain source code of very large and complex projects that may involve multiple programmers. Includes "Branching" to effectively maintain code when programs evolve on multiple paths (e.g., new versions for different systems, or a new program based on an existing program). Single user. **$395**

Network PVCS — Extends Corporate PVCS for use on Networks. File locking and security levels can be tailored for each project. 5-Station License $1,000. Call (503) 645-1150 for pricing on Licenses for more than 5 Stations. **$1000**

TO ORDER: VISA/MC 1-800-547-4000. Dept. No. 350. Oregon and outside US call (503) 684-3000. Send Checks, P.O.s to: POLYTRON Corporation, 1815 NW 169th Place, Suite 2110, Dept. 350, Beaverton, OR 97006.

POLYTRON®

High Quality Software Since 1982

CIRCLE 5 ON READER SERVICE CARD

Stallman criticism

Dear Editor:

This letter is to thank John Heselton for his letter "Brilliant but naive" (Feedback, *COMPUTER LANGUAGE*, May 1987, pp. 9-15). I'm glad someone had the words to defend programmers from the jaws of poverty.

Heselton's letter was in response to a previous article by Judy Getts, "The last true hacker: Richard Stallman" (Computervisions, *COMPUTER LANGUAGE*, May 1987, pp. 99-107). Stallman's belief that programming is a service, not a product, is incorrect. A service cannot be put into a can and sold. A program can. To say that this end product should be free is ridiculous. Why would anyone pay for a product to be developed just to have the next person get it for free?

I agree, Stallman is probably very brilliant. I think if he and I had a war of words, he would probably win. But to say he is naive is putting it mildly. Richard, open your eyes! Money makes the world go around. I know I wouldn't stay with a programming career if all I saw in my future was an hourly wage.

Steven M. Sommers
Orange, Calif.

Wrap-up rebuttal

Dear Editor:

"Fourth generation language data bases" by Richard Finkelstein, William Patterson, and Kasu Sista (Software Reviews, *COMPUTER LANGUAGE*, May 1987, pp. 89-122) was interesting and informative. However, you omitted ORACLE (ORACLE Corp.), the best known and most widely used such product.

In so doing, you rendered the article imcomplete in two ways. First, an objective and thorough review of ORACLE would be more valuable to your readers than one of any other single product. Most of us know its reputation; we would like to know the facts.

Second, ORACLE is as close as you can come to a standard of reference; it would have clarified how all of the products compare to each other. This latter purpose would also have been served by much more detailed tables of product features. For example, how complete are the implementations of SQL? Are nulls and assertions supported?

Finally, although you do not have space to review every product in a given field, you should at least list all of the major ones, compare them in general terms (for example, include them in your feature tables), and indicate why you chose to review the ones you did. I find it strange and arbitrary that there is no mention of PARADOX.

Eric Rosenberg
Greenwich, Conn.

Reviewer Richard Finkelstein responds: ORACLE was identified as a product that should be reviewed, but ORACLE Corp. declined to offer to be reviewed since v. 5.1 of ORACLE was in beta testing. I agree that this product should be considered for a future review.

As I mentioned in the introduction to the wrap-up, we chose products that were non-procedural and set-oriented. PARADOX is highly procedural (except for its query language) and differs little from any of a dozen products on the market with a similar type of language and applications development structure. While PARADOX may provide advantages over its competitors, it is difficult to find anything in the language that can be considered non-procedural or fourth-generation-like.

While I consider a comparison of SQL important, there were several packages reviewed that did not support SQL, and therefore, a comparison was inappropriate for this wrap-up. It may be an interesting topic for a future article.

Quicksort discussion

Dear Editor:

In the past few months, there have been several comments concerning the possibility of implementing Quicksort for linked lists. First, in his response to Bob Gebelein's letter "Unraveling unshuffle" (Feedback, *COMPUTER LANGUAGE*, Jan. 1987, pp. 7-10) Art Kagel states that "Quicksort cannot to my knowledge be implemented for linked lists." In a response to this statement ("Sorting it out," Feedback, *COMPUTER LANGUAGE*, Mar. 1987, pp. 9-11) Steven Kienle states that it could be done if the sort routine first copied the linked list to an array, sorted the array, and then copied the array back to a linked list.

While this is certainly one way to implement Quicksort for linked lists (and possibly the only way if working with a singly linked list), it is not always necessary. It is possible to implement Quicksort so that it works directly with doubly linked lists.

(The implementation I am discussing refers to doubly linked lists that have both a beginning and an end with *NULL* terminators at those points. However, the algorithm can easily be changed, and has been, to work with circular, doubly linked lists.)

There are two major problems with implementing Quicksort for linked lists.

C & PASCAL PROGRAMMERS

Blaise Computing provides a broad range of programming tools for Pascal and C programmers, with libraries designed for serious software development. You get carefully crafted code that can be easily modified to grow with your changing needs. Our packages are shipped complete with comprehensive manuals, sample programs and source code.

C TOOLS PLUS
$175.00
NEW! Full spectrum of general-purpose utility functions; windows that can be stacked, removed, and accept user input; interrupt service routines for resident applications; screen handling including EGA 43-line text mode support and direct screen access; string functions; and DOS file handling.

PASCAL TOOLS/TOOLS 2
$175.00
Expanded string and screen handling; graphics routines; easy creation of program interfaces; memory management; general program control; and DOS file support.

VIEW MANAGER
$275.00
Complete screen management; paint data entry screens; screens can be managed by your application program; block mode data entry or field-by-field control. Specify C or IBM/MS-Pascal.

ASYNCH MANAGER
$175.00
Full featured asynchronous communications library providing interrupt driven support for the COM ports; I/O buffers up to 64K; XON/XOFF protocol; baud rates up to 9600; modem control and XMODEM file transfer. Specify C or IBM/MS-Pascal.

Turbo POWER TOOLS PLUS
$99.95
NEW! Expanded string support; extended screen and window management including EGA support; pop-up menus; memory management; execute any program from within Turbo Pascal; interrupt service routine support allowing you to write memory resident programs; schedulable intervention code.

Turbo ASYNCH PLUS
$99.95
Complete asynchronous communications library providing interrupt driven support for the COM ports; I/O buffers up to 64K; XON/XOFF protocol; and baud rates up to 9600.

RUNOFF
$49.95
NEW! Text formatter written especially for programmers; flexible printer control; user-defined variables; index generation; and general macro facility. Crafted in Turbo Pascal.

EXEC
$95.00
Program chaining executive. Chain one program from another even if the programs are in different languages. Shared data areas can be specified.

ORDER TOLL-FREE 800-227-8087!

BLAISE COMPUTING INC.
2560 Ninth Street, Suite 316 Berkeley, CA 94710 (415) 540-5441

CIRCLE 6 ON READER SERVICE CARD

FINALLY!

EASYFLOW

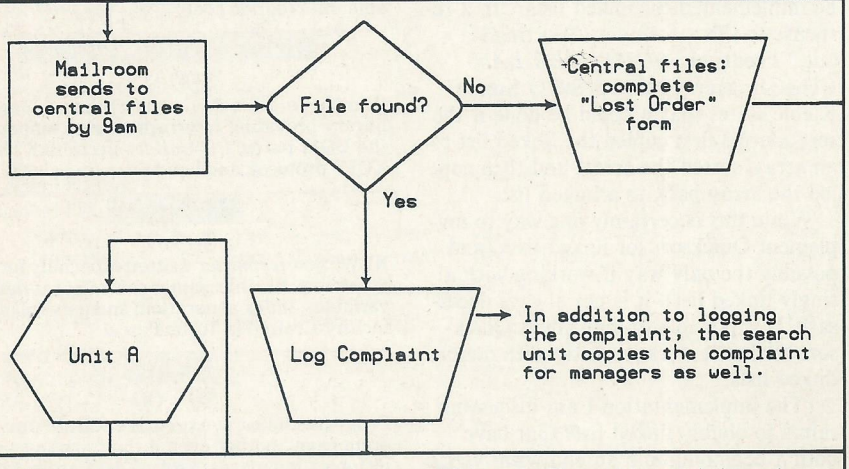
An on-screen flowchart processor that knows about flowcharts - not just another "screen draw" program that makes you do most of the work. EASYFLOW is a powerful full-screen graphics program dedicated to flowcharts and organization charts. With it you can quickly compose charts. More important, you can easily modify charts so they are always up to date.

- **Automatic:** fully automatic text centering within shapes, both horizontally and vertically. Fully automatic line routing.
- **Fast:** written in assembly language for speed.
- **Large:** Charts up to 417 columns wide by 225 lines high. Chart too large for your printer? *EASYFLOW* automatically breaks the chart up & prints it in page size pieces.
- **Standard:** All standard flowcharting shapes included. Custom shapes can be ordered.
- **User friendly:** Don't take our word for it. PC Magazine* says "*EASYFLOW* lives up to its name. It's hard to imagine any easier and more flexible way to produce basic and even complex flowcharts".
- **It prints:** on most popular matrix printers including IBM, Epson, Toshiba, HP LaserJet, LaserJet-Plus and many others.
- **It plots:** on HP7475 and compatible plotters.
- **It works:** we are contractually prevented from mentioning the name of the "big eight" accounting firm that purchased a world-wide site license, but we can tell you that they spent months evaluating all available flowcharting packages before settling on *EASYFLOW*.
- **Rush delivery:** Order by noon today (eastern time) and we'll have it to you by courier tomorrow**. Rush delivery charge is $15.00 (instead of $2.00) and is available only in USA & Canada.
- **Documented:** (100+ page) manual and over 150 screens of context sensitive help.

EASYFLOW works on IBM PC's and compatibles. Requires 320 K memory, DOS 2.0 or higher and an IBM CGA, IBM EGA or Hercules monochrome adapter card. Order direct for only **$149.95** + $2.00 S&H (USA/Canada), $10.00 (foreign). Payment by M.O., cheque, VISA, Mastercard or Company PO.

* March 10, 1987 issue, page 278.

** Rush orders are shipped by Purolator Courier and normally arrive the next business day to most locations. Remote destinations take longer.



The chart fragment above was produced on an HP LaserJet-Plus and is actual size and unretouched. Publication quality charts like this can be produced using only minutes of preparation time and seconds of print time.

HavenTree Software Limited
PO Box 1093-K
Thousand Island Park, NY 13692

Order Desk: 1-800-267-0668
Information: (613) 544-6035 ext. 46
Telefax(G3): (613) 544-9632

CIRCLE 57 ON READER SERVICE CARD

First, the sort must be able to traverse the list both from beginning-to-end and end-to-beginning. Second, the sort needs to know when the pointer list from beginning-to-end passes the pointer traversing the list from end-to-beginning.

The first problem is easily solved by the use of a doubly linked list. At first glance the second problem would seem to preclude using Quicksort on linked lists. After all, how does the sort know when two pointers have passed each other in relative position within a linked list?

It turns out that solving the second problem is not really very difficult. Essentially, the sort routine logically maps array index numbers to the linked list elements. To do this, it needs only two additional, easily attainable items of information about the list: the relative index numbers of the beginning and ending elements of the list. These are passed, along with pointers to the beginning and ending elements of the list, to the sort routine. The routine matches each index with its corresponding pointer and, whenever a pointer is assigned to the next or previous list element, the corresponding index is incremented or decremented, respectively. By doing this, the test to see if the two pointers have passed each other becomes a simple relational test between the two indexes.

Quicksort can be, and is, implemented for doubly linked lists. The implementation given here has proven very useful in the projects I have worked on. It is easily extensible to secondary and other sort keys. Also, by using multiple sets of forward and backward links and making some modifications to the code, multiple sort orders of the list can be maintained while keeping only one copy of the data. This is especially useful when the data may be sorted and resorted several different times by an application (as when displaying tabular data to a user).

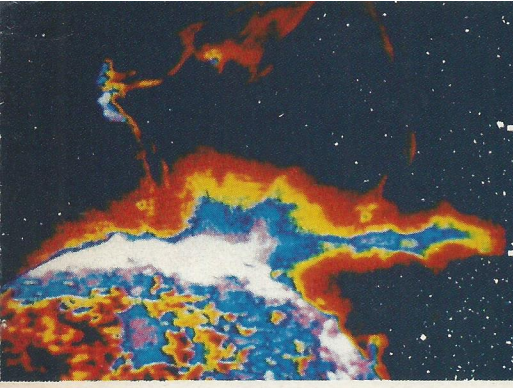
Mike Higginbotham
Plano, Texas

Clarifications

Dear Editor:

At last! Joseph Sant's "A Simple Data Compression Technique" *COMPUTER LANGUAGE,* Mar. 1987, pp. 65-66) is an article fully comprehensible even to me. Sant's text compression utility is easily understood, small-sized, and works most rapidly indeed.

Since I am a bit of a novice with any language, especially with C, my most recent, it came as something of a surprise that the compression algorithm is just a fraction too simple. Like most other typists, I cannot overcome the habit of pressing the space bar twice at the end of each sentence. Word processors that jus-



E=MC² AZTEC

Our thanks to NASA for supplying this computer enhanced ultraviolet photo taken by Skylab IV of a solar prominence reaching out 350,000 miles above the sun's surface

Genius Begins With A Great Idea ...

But The Idea Is Just The Beginning

What follows is the time consuming task of giving form and function to the idea.

That's why we concentrate on building into our software development systems functions and features that help you develop your software ideas in less time and with less effort.

We've started 1987 by releasing new versions of our MS-DOS, Macintosh, Amiga, ROM, and Apple // C development systems. Each system is packed with new features, impressive performance, and a little bit more genius.

Aztec C86 4.1

New PC/MS-DOS • CP/M-86 • ROM

Superior performance, a powerful new array of features and utilities, and pricing that is unmatched make the new Aztec C86 the first choice of serious software developers.

Aztec C86-p Professional System $199
• optimized C with near, far, huge, small, and large memory + Inline assembler + Inline 8087/80287 + ANSI support + Fast Float (32 bit) + optimization options • Manx Aztec 8086/80x86 macro assembler • Aztec overlay linker (large/small model) • source level debugger • object librarian • 3.x file sharing & locking • comprehensive libraries of UNIX, DOS, Screen, Graphics, and special run time routines.

Aztec C86-d Developer System $299
• includes all of Aztec C86-p • Unix utilities make, diff, grep • vi editor • 6+ memory models • Profiler.

Aztec C86-c Commercial System. $499
• includes all of Aztec C86-d • Source for library routines • ROM Support • CP/M-86 support • One year of updates.

Aztec C86 Third Party Software

A large array of support software is available for Aztec C86. Call or write for information. The following is a list of the most requested products: Essential Graphics • C Essentials • C Utility Library • Greenleaf Com. • Greenleaf General • Halo • Panel • PC-lint • PforCe • Pre-C • Windows for C • Windows for Data C terp • db Vista • Phact • Plink86Plus • C-tree.

CP/M • TRS-80 • 8080/Z80 ROM

C compiler, 8080/Z80 assembler, linker, librarian, UNIX libraries, and specialized utilities.

Aztec C II-c (CP/M-80 & ROM). $349
Aztec CII-d (CP/M-80) $199
Aztec C80 (TRS-80 3&4) $199

Aztec C68k/Am 3.4

New Amiga Release

Amiga user groups across the USA voted Aztec C68k/Am release 3.3 the best Software Development System for the Amiga. Release 3.4 is more impressive.

Aztec C68k/Am-p Professional $199
A price/feature/performance miracle. System includes: optimized C • 68000/680x0 assembler • 68881 support • overlay linker • UNIX and Amiga libraries • examples.

Aztec C68k/Am-d Developer $299
The best of Manx, Amiga, and UNIX. System includes: all of Aztec C68k/Am-p • the Unix utilities make, diff, grep and vi.

Aztec C68k/Am-c Commercial $499
Aztec C68k/Am-d plus source for the libraries and one year of updates.

Aztec C68k/Mac 3.4

New Macintosh Release

For code quality, reliability, and solid professional features, Aztec C for the Macintosh is unbeatable. This new release includes features and functions not found in any other Macintosh C development system.

Aztec C68k/Mac-p Professional $199
• optimized C • 68000/680x0 assembler • 68881 support • overlay linker • UNIX and Macintosh libraries • examples.

Aztec C68k/Mac-d Developer. $299
The best of Manx, Macintosh, and UNIX. System includes: all of Aztec C68k/Am-p • the Unix utilities make, diff, grep • vi editor.

Aztec C68k/Mac-c Commercial $499
Aztec C68k/Am-d plus source for the libraries and one year of updates.

Aztec C65

New ProDOS Release

Aztec C65 is the only commercial quality C compiler for the Apple II. Aztec C65 includes C compiler, 6502/65C02 assembler, linker, library utility, UNIX libraries, special purpose libraries, shell development environment, and more. An impressive system.

Aztec C65-c Commercial $299
• runs under ProDOS • code for ProDOS or DOS 3.3

Aztec C65-d Developer. $199
• runs under DOS 3.3 • code for DOS 3.3

Aztec ROM Systems

6502/65C02 • 8080/Z80 • 8086/80x86 • 680x0

An IBM or Macintosh is not only a less expensive way to develop ROM code, it's better. Targets include the 6502/65C02, 8080/Z80, 8086/80x86, and 680x0.

Aztec C has an excellent reputation for producing compact high performance code. Our systems for under $1,000 outperform systems priced at over $10,000.

Initial Host Plus Target $750
Additional Targets $500
ROM Support Package $500

Vax, Sun, PDP-11 ROM HOSTS

Call for information on Vax, PDP-11, Sun and other host environments.

C' Prime

PC/MS-DOS • Macintosh
Apple II • TRS-80 • CP/M

These C development systems are unbeatable for the price. They are earlier versions of Aztec C that originally sold for as much as $500. Each system includes C compiler, assembler, linker, librarian, UNIX routines, and more. Special discounts are available for use as course material.

C' Prime $75

Aztec Cross Development Systems

Most Aztec C systems are available as cross development systems. Hosts include: PC/MS-DOS, Macintosh, CP/M, Vax, PDP-11, Sun, and others. Call for information and pricing.

How To Become An Aztec C User

To become a user call 800-221-0440. From NJ or international locations call 201-542-2121. Telex: 4995812 or FAX: 201-542-8386. C.O.D., VISA, MasterCard, American Express, wire (domestic and international), and terms are available. One and two day delivery available for all domestic and most international destinations.

Aztec C is available directly from Manx and from technically oriented computer and software stores. Aztec Systems bought directly from Manx have a 30 day satisfaction guarantee.

Most systems are upgradable by paying the difference in price plus $10. Site licenses, OEM, educational, and multiple copy discounts are available.

MANX

Manx Software Systems
1 Industrial Way, Eatontown, NJ 07724

To order or for more information call today.

1-800-221-0440

In NJ or international call (201) 542-2121 • TELEX: 4995812

MS is a registered TM of Microsoft, Inc., CP/M TM DRI, HALO TM Media Cybernetics, PANEL TM Roundhill Computer Systems, Ltd., PHACT TM PHACT Assoc., PRE-C, Plink-86, Plink-86+, P-Force TM Phoenix, db Vista TM Raima Corp., C-terp, PC-lint, TM Gimpel Software, C-tree TM Faircom, Inc., Windows for C, Windows for DATA TM Creative Solutions, Apple II, Macintosh TM Apple, Inc., TRS-80 TM Radio Shack, Amiga TM Commodore Int'l., Unix TM AT&T, Vax TM DEC, Aztec TM Manx Software Systems.

CIRCLE 7 ON READER SERVICE CARD

Finally, powerful graphics for Turbo Pascal!

Announcing:

TurboHALO Graphics!

- 150 graphic primitives (based upon the powerful HALO primitives).
- Graphics in up to 16 colors, in medium or high resolution, on 10 popular graphics cards—including **IBM CGA and EGA, Hercules, and AT&T DEB.**
- Output to over 20 printers and plotters—including **HP and Corona Laser and Apple Imagewriter.**
- 11 popular input devices supported.
- Memory resident drivers require only 2K of Turbo Pascal code space.
- These professional graphics programs use **HALO Primitives:**

  Drafix II—Foresight Resources
  CADKEY—Micro Control Systems, Inc.
  Dr. Halo II—Media Cybernetics/IMSI
  CAD Master—Datagraphics
  Decision Master—Centec
  Artworks—West End Films

- Speed, ease of use, multiple fonts and much more, for **ONLY $129.00! NO ROYALTY FEES!**

COMPARE THE FEATURES!

TurboHALO vs. Borland's Turbo Grafix ToolBox:

Feature	TurboHALO	Borland
16-Color Support?	**Yes**	No!
Full EGA Support?	**Yes**	No!
Laser Printer Support?	**Yes**	No!
Multiple Input Drivers?	**Yes**	No!
Do Professional Programs Use Same Primitives?	**Yes**	No!

Call Now To Order!

IMSI™

1299 Fourth Street • San Rafael, California 94901 • (415) 454-7101
Toll Free (800) 222-GRAF (Outside CA) • **(800) 562-GRAF** (In CA)

TURBOHALO is published by IMSI and is a registered trademark of MEDIA CYBERNETICS, INC.

CIRCLE 43 ON READER SERVICE CARD

tify text insert many extra spaces between words. The program as printed will delete only the first space, regardless of how many follow it.

The addition of one extra variable, *next*, and and a short *while* loop cures this problem:

```
...
if ( c == ' ')
   {
      prev_chr = 128;
      while (( next = getc(infile))
         == ' '); /* do nothing */
      c = next;
   }
...
```

The *while* loop simply gobbles every space after the first, terminating when it hits a non-space character.

Frank Slavin
Rural Retreat, Va.

[redacted] license required

In the general information table that appeared in "Moving Applications from dBASE to C" (COMPUTER LANGUAGE, *May 1987, pp. 67-73) under the heading "One-time cost—Royalty," there was an incorrect entry for Computer Control Systems Inc. Application programmers using products in programs for resale from the company do need to obtain a software license agreement as stated.—Ed.*

[redacted] Finding ddb

The ddb UNIX debugger mentioned in David Livshin's "A Source-Level Symbolic Debugger for UNIX" (COMPUTER LANGUAGE, *June 1987, pp. 53-56) is marketed by Charles River Data Systems as esdb (extended symbolic debugger). The product is part of UNOS, a UNIX V real-time operating system.*

For more information contact Dick Swee at Charles River Data Systems, 983 Concord St., Framingham, Mass., (617) 626-1111.—Ed.

Introducing 1st-CLASS Fusion™

The power of 1st-CLASS. Plus a new level of software and system connectivity.

Ever since the initial appearance of **1st-CLASS**, the A.I. community has hailed it as one of the most powerful, easy-to-use IBM-PC packages on the market.

Now we've made it even better. Both internally and—even more important—externally.

On the inside, we've added a range of new features to create **1st-CLASS** version 3.5. These features include multiple help/reference windows, fill-in-the-blank data entry forms, a full session report generator, and an "Instant Replay" capability that lets you go back to your last session exactly as you left it.

On the outside, we've done even more. With **1st-CLASS Fusion**, we've made possible virtually unlimited connectivity between **1st-CLASS Fusion** and other hardware and software systems.

- Our new code generators not only save programming time, but allow you to transport your work anywhere you like— all the way to a mainframe.
- Our graphics capture and display utilities make it as easy as 1,2,3 to capture screens from other programs and display them within an advisor.
- Our math calculator lets you incorporate math operations into knowledge bases, then pass them on to other programs.
- Our dBASE III interface lets you access .DBF files directly from a running advisor.
- And our road map helps you build large, chained expert systems—then move freely among them at a single keystroke.

The result is a system that's easier to use than ever before—in more ways, and in more places. Which is just what you'd expect from Programs in Motion Inc.: the company that brought you **1st-CLASS** in the first place.

Find out for yourself why AT&T, Citibank, duPont, 3M, and hundreds of others have all chosen to go **1st-CLASS**.

Call toll free today for more information:

1st-CLASS Fusion, $1295 ($995 for current 1st-CLASS owners);
1st-CLASS version 3.5, $495 (upgrades from 1st-CLASS 3.0, $50);
1st-CLASS demo package, $20.

What they say about 1st-CLASS

"For every application that we can imagine, 1st-CLASS is clearly the best program reviewed here."
—Social Sciences Microcomputer Review, Summer, 1986

"1st-CLASS is among the best expert systems available for personal computers. The developer obviously cares about its users (this newest version is largely in response to user suggestions) and that care shows."
—ONLINE Today, Dec. 1986

"1st-CLASS is probably the most popular inductive tool available, and it is certainly one of the easiest to use."
—Paul Harmon, Expert System Strategies, Dec. 1986

"The documentation is a work of art. I doubt if Programs in Motion gets many technical assistance calls."
—The Computer Shopper, March 1987

"It provides an easy way to implement expert systems at a price that won't break the bank."
—Computer Language, March 1987

To order or for more information, call toll-free today!

1-800-872-8812

PROGRAMS IN MOTION INC.

10 Sycamore Road
Wayland, Massachusetts 01778

1-617-653-5093 (in MA)

1st-CLASS and 1st-CLASS Fusion: Easy-to-use, easy-to-connect expert systems.

☐ 1st-CLASS Fusion	$1295.
☐ 1st-CLASS	495.
☐ Demonstration Package	20.
☐ Literature	Free

MA residents add 5% sales tax.
Shipping Charge: Canada—$20; all other foreign countries—$50.
Programs in Motion Inc.
10 Sycamore Road, Wayland, MA 01778
1-800-872-8812
In MA call: 1-617-653-5093

NAME

COMPANY

TITLE PHONE #

ADDRESS

CITY, STATE, ZIP

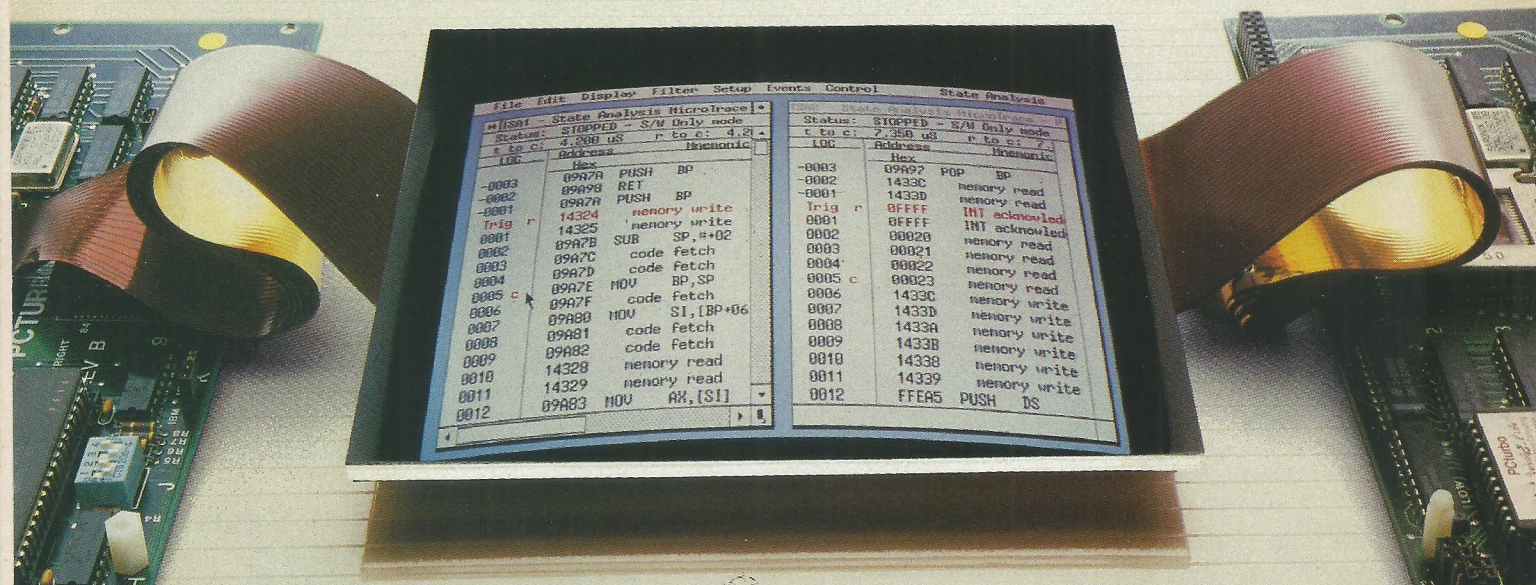
CREDIT CARD # EXP. DATE ☐ MC ☐ VISA

© 1987 Programs in Motion Inc.

dBase® is a registered trademark of Ashton-Tate, Inc.
IBM® is a registered trademark of International Business Machines Corporation.

CIRCLE 9 ON READER SERVICE CARD

At last, multi-processor time-aligned trace.



Since so many embedded microcomputer designs employ more than one processor, you'd think there would be many tools to trace multi-processor dialog in both assembly-level and high-level code. And then time-align the results.

In fact, there's only one. The Software Analysis Workstation™ (SAW) from NWIS.

The SAW alone gives you a choice of a time-aligned, assembly-level trace on both processors. Or a time-aligned trace of module/procedure events on one processor and assembly-level activity on a second processor.

At last, you have a fast, effective means for evaluating complex, inter-processor transactions as they occur in real-time in the target system.

The SAW's dual-processor trace is displayed in a fully time-synchronized manner, so you can instantly analyze relationships between the two execution streams. Scroll the trace from one processor, and the trace from the other processor can be locked to scroll right along with it. The timing relationship between the two traces automatically stays intact, even if the two processors have different clock frequencies.

And multi-processor, time-aligned trace is only one of the SAW's outstanding debug, optimization and verification functions. You also get real-time, non-intrusive performance analysis. And CodeMap,™ for rapid and complete verification of code coverage. All these tools are compiler/assembler and host development computer independent.

The SAW is already producing dramatic results for software engineers around the world. Let us connect the SAW to your target system and you'll see why. For an on-site demo and free applications library, phone: 1-800-547-4445

NWIS

NORTHWEST INSTRUMENT SYSTEMS, INC.

P.O. Box 1309 • Beaverton, OR 97075 • 1-800-547-4445

AUSTRIA: 0222-253626 **BELGIUM:** 02-2418130 **DENMARK:** 05-611100 **FINLAND:** 0-5284316 **FRANCE:** 014-5347535 or 016-9070935 **ISRAEL:** 03-403373 **ITALY:** 02-470646 **THE NETHERLANDS:** 01720-43221 **NORWAY:** 02-789460 **SPAIN:** 01-4558112 **SWEDEN:** 08-7390045 **SWITZERLAND:** 01-7231410 **UNITED KINGDOM:** 06284-4426 **NWIS-USA:** 503-690-1300 **WEST GERMANY:** 089-858020

CIRCLE 10 ON READER SERVICE CARD

PROGRAMMING ON PURPOSE

Who's the boss?

By P.J. Plauger

Method:
Top-down design

Description:
Top-down design focuses on the decomposition of a program into a hierarchy of modules. It can be viewed either as a process of refinement by steps, from the most general to the specific; or it can be viewed as a technique for mapping a data flow network into a hierarchy of modules. The name derives from the convention of drawing hierarchical decompositions of functions with the most general modules (the root of the tree) at the top and the most detailed modules (the leaves) at the bottom.

Mapping data flow networks is the safer way to evolve a hierarchy of modules, because it offers several models for proven solutions based on frequently occurring network topologies.

Documentation:
A data flow diagram retains the most information about a modular decomposition. At the procedural level, a structure (or HIPO) chart documents the modules and data interfaces of a given hierarchy.

Limitations:
Without a clear understanding of the flow of data at the appropriate level of abstraction, top-down design offers little guidance and leads to an excess of modules. Recursion can tie knots in a structure chart.

When I discussed bottom-up design several months ago ("Finite-state machines," *COMPUTER LANGUAGE*, Oct. 1986, pp. 15-19), I annoyed a number of people by focusing on the construction of finite-state machines. I know several other design methods fall under that name, but I didn't hesitate to apply the name to finite-state machines alone.

This month, I expect to annoy even more people by discussing top-down design almost purely in terms of data flow analysis. I know that for many readers top-down design is a blanket term that covers any of several right ways to do things. It is stepwise refinement, it is prototyping, it is goal-directed programming. I am willing to stipulate that top-down design is indeed all of these things, provided you are willing to concede that more guidance may be needed to turn an approach into a design method.

In my experience, the greatest guidance in elaborating a top-down design comes from an analysis of the flow of data through the program you are trying to build. The goal of top-down design is to produce a list of all the functions (routines, procedures, subroutines) required to perform a set of operations within one computer program.

In a computer program, one main function takes control when the program starts up. In turn, the main function calls all of the other functions, exchanging data in the form of arguments, function return values, and access to shared global data. At this level of abstraction the list of functions should be accompanied by:

- A list of the data passed into each function when it is called.
- A list of the data passed back from the function when it is called.
- A list of the shared global data that may be accessed or modified by the function when it is called.
- A brief (often one sentence) description of what each function does.
- A list of the functions called in turn by each function.

Essentially, all of this data can be captured on a diagram called a structure chart. IBM popularized a more stylized version of the structure chart, a Hierarchy with Input, Process, and Output (HIPO) chart.

Figure 1 shows a small structure chart. Each box represents a function, each large arrow a potential call to that

Simple structure chart

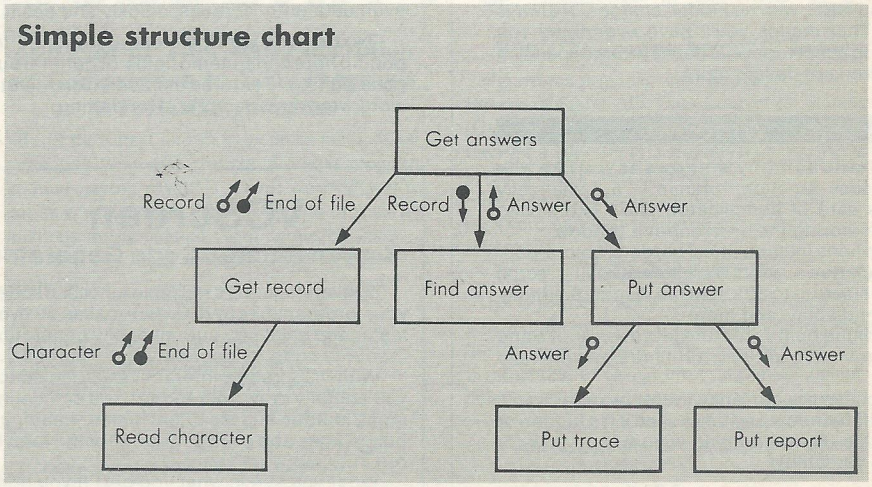


Figure 1.

Version 3.0

New Features

Windows, Data Entry, Help Management, Menus, Text Editing, plus ...

SOURCE CODE

FREE

Vitamin C

It's good for your system!

The Vitamin C Difference

With **Vitamin C,** your applications come alive with windows that explode into view! Data entry windows and menus become a snap, and context sensitive pop-up help messages are nearly automatic.

With **VCScreen,** you'll save time by interactively painting windows and forms so what you see is what you get! Then, one button generates C source code ready to plug into your program and link with Vitamin C.

Easy enough for the beginner. Versatile enough for the professional. Vitamin C's **open-ended design** is full of "hooks" so you can intercept and "plug-in" special handlers to customize or add features to most routines.

Of course, Vitamin C **includes all source code FREE,** with no hidden charges. *It always has.* That means you'll have everything you need to adapt to special needs without spending hundreds of dollars more.

Windows

Create as many windows as you like with one easy function. Vitamin C automatically takes care of complicated tasks like saving and restoring the area under a window.

Options include titles, borders, colors, pop-up, pull-down, zoom-in, 4-way scrolling, scroll bars, sizes up to 32k, text file display & editing, cursor display, and more.

Unique built-in feature lets users move and resize windows during run-time via a definable key.

Access the current window by default or a specific window any time, even if it's hidden or invisible. Save and load windows on disk for more versatility!

Data Entry

Flexible dBase-like data entry and display routines feature protected, invisible, required, and scrolling fields. Picture clause formatting, full color/attribute control, selection sets, single field and full screen input, and unlimited data validation via standard and user definable routines. That means you aren't locked into one way of doing things.

Vitamin C even provides true right-to-left input of numeric fields with dynamic display of separators & currency symbols.

High Level Functions

Use our intergrated help management, multi-level menus, and text file routines, or build your own handlers using Vitamin C's basic windowing and data entry routines.

Standard help handler provides context sensitive pop-up help messages any time the program awaits key strokes. The help text file is stored on disk and indexed for quick access. So easy to use that a single function initializes & services requests by opening a window, locating, formatting, displaying, and paging through the message.

Multi-level "MacIntosh" & "Lotus" style menus make user interfaces and front ends a snap. Menus can call other menus, functions, even data entry screens, quickly and easily.

Text editor windows can be opened for pop-up note pads, memo fields, or general purpose editing. Features include insert, delete, word wrap, and paragraph formatting.

VCScreen

Screen Painter/Code Generator

Just as Vitamin C's reusable functions speed your programming, VCSreen makes it even *faster and easier* by automatically generating C source code for your data entry screens!

With VCScreen's interactive screen editor, you actually draw your forms. You can define input, output and constant fields, headings, boxes, lines and even a window for the form to run in.

What you see is what you get. If you don't like the position of an object, just "pick it up" with the cursor and move it! Changing colors, attributes, copying, and deleting is just as easy.

VCScreen generates readable C source code. It declares variables with names you provide and can even generate structures.

With VCScreen choosing the right functions, parameters and sequences, and Vitamin C supplying the functions to choose from, you can stop worrying about semi-colons, matching braces, and calling conventions and concentrate on creating your application!

30 Day Money Back Guarantee

Better than a brochure. More than a demo disk. If you're not satisfied, simply return the package within 30 days and receive a full refund of the purchase price.

Vitamin C $225.00

Includes ready to use libraries, tutorial, reference manual, demo, sample, and example programs, and quick reference card. For IBM PC and compatibles. Specify Microsoft, Lattice, Computer Innovations, Aztec, Mark Williams, Wizard, DeSmet, or Datalight C compiler AND compiler version number when ordering.

Vitamin C Source ... FREE*

**Free with purchase of Vitamin C*

VCScreen $99.95

Requires Vitamin C and IBM PC/XT/AT or true compatible.

ALL ORDERS:

SHIPPING: $3 ground, $6 2-day air, $20 overnight, $30 overseas. Visa and Master Card accepted. All funds must be U.S. dollars drawn on a U.S. Bank. Texas residents add 7¼% sales tax.

For Orders or More Information, Call ...

(214) 416-6447

creative PROGRAMMING

Creative Programming Consultants, Inc.
Box 112097 Carrollton, Texas 75011

function from another function. Each small arrow is a data item that is read (down arrow) or written (up arrow) by the called function. As you can see, you can pack quite a lot of useful information into a structure chart.

The structure chart is an excellent document for capturing the initial design of a program. You have something concrete that is detailed enough to evaluate yet abstract enough to keep you out of the mire of implementation details. It is close enough to the everyday experience of writing code that many of us programmers can evaluate the likely success of the coding effort just from the data flow and function descriptions. Design quality can also be measured by a number of criteria that are applicable only after a structure chart has been generated. (They will be the subject of a future column.)

You are well-advised to maintain the structure chart throughout the lifetime of a program. Changes do occur in the number and nature of functions that constitute a program, but such changes are fewer than those that leave the structure chart unchanged. The effort of keeping the structure chart current is well repaid because you have a quick overview of the program and a handy road map to the functions and data that are likely to be affected by change. If the structure chart for a program is at all nontrivial, it is well worth generating and keeping alive.

So how do you go about making this marvelous document? The easiest way is to write the program, get it working, then document what you did. That puts the structure chart in the same category as, say, a cross reference of all identifiers defined in your program. It helps with program maintenance, but not with design. If you want to generate a structure chart as part of the design creation and evaluation process, however, you need a bit more guidance.

The HIPO approach is one attempt to add guidance. It tells you to write the main function box at the top of the page and describe its function the same way you would describe the overall function of the program. This description could be "produce graphs," or "update data base from transactions," or "perform Fourier analysis on spectra." You then identify three subfunctions that:

- Obtain input at the appropriate level of abstraction.
- Process the input to produce output.
- Deliver the output.

For each of these subfunctions, you then apply the same paradigm. For input functions, you ask yourself what three subfunctions are appropriate to obtain (somewhat less refined) input, do the process, and generate the (refined) output. For output functions, you ask yourself what three subfunctions will result in the production of somewhat less-refined output. And for process functions, you're on your own.

Eventually, you should arrive at a level where input/output can be performed in terms of the I/O services of the programming language you are using, and where all processing can be performed in terms of statements in the language or calls to standard library functions.

In real life, you probably discover the need for a few common subfunctions along the way and stop what you're doing long enough to go specify them. This is a way of bringing the bottom up to meet you, as it were, while you're mostly progressing top down. Nothing to be ashamed of—everybody does it.

Also, you shouldn't be too religious about decomposing each module into three subfunctions: input, process, and output. Often, one or two of these subfunctions becomes trivial long before you get to the bottom-most module for the third. If you find yourself making up silly little functions just to keep the hierarchy balanced, then you've fallen into the HIPO trap. As we shall see, there is no particular virtue in having symmetric structure charts. More fundamental forces are at work, shaping the hierarchy of a program.

A more important limitation of this approach to top-down design, as I have said before ("Which tool is best?," *COMPUTER LANGUAGE*, July 1986, pp. 15-19), is that it's easy to get lost on the way down. If the problem is small, you can usually succeed—but then the design method doesn't help all that much. If the problem is familiar, you can also succeed—but then you are primarily imitating a past design. What you need for big-league problems is a set of guidelines that let you:

- Make small problems out of big ones.
- Identify familiar problems within unfamiliar ones.
- Organize the smaller pieces into a coherent whole.

This is where data flow analysis moves to the fore in top-down design. With this approach, you first develop a data flow diagram for the problem to be solved. You then derive the structure chart al-

FULL AT&T C++ for half the price of our competitors!

Guidelines announces its port of **version 1.1** of AT&T's C++ translator. As an object-oriented language, C++ includes: classes, inheritance, member functions, constructors and destructors, data hiding, and data abstraction. 'Object-oriented' means that C++ code is more readable, more reliable and more reusable. And that means faster development, easier maintenance, and the ability to handle more complex projects. C++ is **Bell Labs' answer to Ada and Modula 2**. C++ will more than pay for itself in saved development time on your next project.

C++

from GUIDELINES for the IBM PC: $195

Requires IBM PC/XT/AT or compatible with 640K and a hard disk.
Note: C++ is a *translator*, and requires the use of Microsoft C 3.0 or later.

Here is what you get for $195:

- The full AT&T v1.1 C++ translator.
- Libraries for stream I/O and complex math.
- "The C++ Programming Language", the definitive 327-page tutorial and description by Bjarne Stroustrup, designer of C++.
- Sample programs written in C++.
- Installation guide and documentation.
- 30 day money back guarantee.

To order:
send check or money order to:

GUIDELINES SOFTWARE
P.O. Box 749
Orinda, CA 94563

To order with Visa or MC, phone (415) 254-9393.
(CA residents add 6% tax.)

C++ is ported to the PC by Guidelines under license from AT&T.
Call or write for a free C++ information package.

CIRCLE 11 ON READER SERVICE CARD

most mechanically from the data flow diagram. The hardest part of the design, in fact, is hidden in the word "almost."

Of the many structure charts you can derive from a given data flow, you need some guidance to determine which is best. A data flow diagram is nonprocedural. It deals with the connectedness of the problem in terms of the data transformations that must occur, without breaking the problem into a hierarchy of functions. It is a natural by-product of the analysis phase before program packaging decisions come into play. It is often easier for civilians to understand a data flow diagram well enough to perform a constructive review than for them to understand the procedural description that implements the data flow.

I talked quite a bit about how to refine data flow diagrams in the last two columns on outside-in design ("Marrying data structures," *COMPUTER LANGUAGE*, May 1987, pp. 17-22, and "Divorcing data structures," *COMPUTER LANGUAGE*, June 1987, pp. 19-24).

In those columns I focused on breaking the data flow transforms into subnetworks of simpler transforms. Each simpler transform can be implemented with a single function whose structure is dictated by the structure of the data streams flowing in and out of the transform. You should always apply outside-in design repeatedly to a data flow diagram, before you begin top-down design, until you can make no further improvements.

Last month, I stopped just short of showing how to turn data flow diagrams into structure charts. Now you are ready to learn the great secret.

All you do is grab the data flow diagram by one bubble and shake it hard. The result, every time, is a structure chart that you can implement, does the job, and has the minimum connectedness required by the problem to be solved.

Figure 2 shows a typical data flow diagram that can result from data flow

All you do is grab the data flow diagram by one bubble and shake it hard

analysis plus the application of outside-in design. The double arrow indicates the module I chose as the grabbing point (C). With a flick of the wrist, Figure 3 results.

Outside-in design assures that each box in the structure chart performs one function that can be implemented by code whose structure captures the structure of the data flowing in and out (as arguments or calls to subfunctions). Data flow analysis assures that all necessary data sharing, and no more, is present. This is not just a top-down design, it is probably a good one.

I haven't told the whole story, of course. I gave no justification for my choice of bubbles to grab in Figure 2. The one I picked gave a nice-looking structure chart, but who is to say that it is the best? I could have picked the bubble below and to the left of my actual choice (E), or the one below that (F). Indeed, I have seen programs based on those choices, and they work. I could even have picked one on the fringes and make any of them work as well.

Figure 4 shows the structure chart that results if you choose the fringe module at the lower left of the diagram (G). Such modules are seldom chosen in real life to serve as the main function of a program hierarchy, but people seldom explore why that is so.

Believe it or not, you can truly pick up a data flow diagram by any of its bubbles, shake it out, and write a program that works from the resultant structure chart. What you find, however, is that some of the structure charts require more exotic coding than others to get the data to appear in the right place at the right time.

Data flow diagrams with merge points, like the two in Figure 2, cause one set of problems. Merges are reasonably well understood but still easy to get wrong. Most people automatically make a single merge point the main function of a program, and with good reason, as it turns out.

Data flow diagrams that contain loops are often even more troublesome because you have data that both merges and diverges at different points in the diagram. Designers usually fall back on various ad hoc structures to implement such data flows, and they often introduce an excess of data coupling between modules as a result.

So it turns out that all of the bubbles in a data flow diagram are equally valid candidates to serve as the main function of a program hierarchy, but some are more equal than others—just like the pigs and other animals in George Orwell's *Animal Farm*.

The data flow diagram will clearly show what data must flow to what modules to get the job done. What the data flow diagram doesn't tell you is how to coordinate the flow of data between modules. That's where you have to get procedural. And it is the procedural criteria, in the end, that determines who

Typical data flow diagram

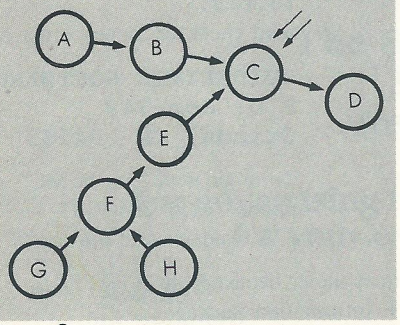


Figure 2.

Structure chart from Figure 2

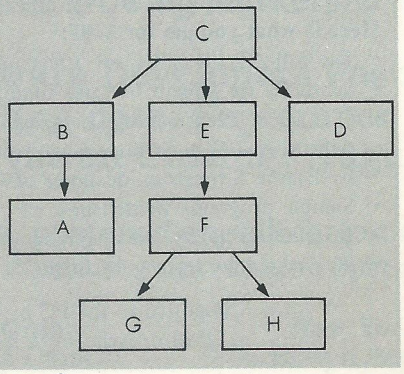


Figure 3.

Another structure chart from Figure 2

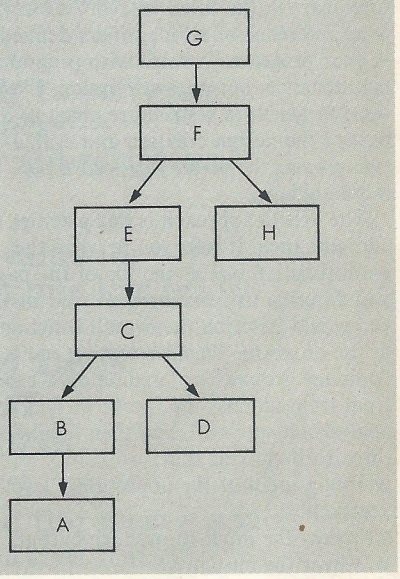


Figure 4.

9-Track Tape Subsystem

for the IBM PC/XT/AT
XENIX or
MS-DOS.

The solution to your micro/mainframe communications problem is available today!

Qualstar's new ½ inch 9-track MINISTREAMER™ brings full ANSI data interchange capability to the PC. Now you can exchange data files with virtually any other computer using 9-track tape.

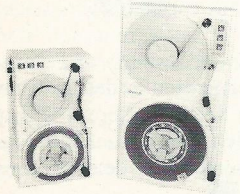
Available in both 7" and 10½" versions, the MINISTREAMER weighs in at only 27 pounds and uses less desk space than an ordinary sheet of paper, yet provides full 1600/3200 BPI capability at an affordable price. Up to 134 megabytes of data (depending on format) can be stored on a standard 10½" reel of tape, thus making the MINISTREAMER a highly-reliable answer to your backup requirements as well.

Tape subsystem includes tape drive, coupler card, cables, dust-cover and MS-DOS or XENIX compatible software. Prices start at $2,995.

386 READY!

Discover the many advantages 9-track tape has over other Micro/Mainframe links.

Call us today!



QUALSTAR®

9621 Irondale Avenue,
Chatsworth, CA 91311
Telephone: (818) 882-5822

CIRCLE 12 ON READER SERVICE CARD

FLASH UP YOUR PROGRAMS WITH

POPSCREEN™

SCREEN GENERATOR / WINDOW MANAGER

INTEGRAL WINDOWS FOR YOUR LANUGAGE
Simple procedure calls open & close any number of windows.

NO RESIDENT LOADERS - NO SATELITE DISPLAY FILES
PopScreen compresses your windows, menus, & displays into compact library modules, inline code, or .bin files customized to your language. (Average size only 100 - 300 bytes) It all links integrally into your program at compile or link time..

POWERFUL BUT SIMPLE DESIGNING PROGRAM - Design your displays and windows onscreen, while you see them. PopScreen gives you easy boxes, complete character graphics control, block moves, superimpose, boilerplating, etc. Manual, ten minute tutorial, and sample programs included.

INSTANT SCREEN LOADING - Professional looking windows open and close simply. Displays pop to the screen in .015 seconds. Your screenwork is easy, fast, and versatile with PopScreen.

POPSCREEN INSTALLS TO SUPPORT:
C: IBM, MICROSOFT, LATTICE (all memory models)
ASSEMBLER (PopScreen will output assembly code)
PASCAL: TURBO (inline code), **IBM, MICROSOFT**
QUICKBASIC (library modules); **DBASE III+** (loadable .bin files)

POPSCREEN, only $39.95
BaySoft, Box 6562-L, Albany, Ca. 94706 415-527-3300

30 DAY SATISFACTION GUARANTEE

CIRCLE 13 ON READER SERVICE CARD

UNIX TOOLS FOR YOUR PC

PC/VI™

UNIX's VI Editor Now Available For Your PC!

Are you being as productive as you can be with your computer? An editor should be a tool, not an obstacle to getting the job done. Increase your productivity today by choosing **PC/VI**—a COMPLETE implementation of UNIX* VI version 3.9 (as provided with System V Release 2).

PC/VI is an implementation of the most powerful and most widely used full-screen editor available under the UNIX operating system. The following is only a hint of the power behind **PC/VI:**

- Global search or search and replace using regular expressions
- Full undo capability
- Deletions, changes and cursor positioning on character, word, line, sentence, paragraph, section or global basis
- Editing of files larger than available memory
- Shell escapes to DOS
- Copying and moving text
- Macros and Word abbreviations
- Auto-indent and Showmatch
- MUCH, MUCH MORE!

Don't take it from us. Here's what some of our customers say: "Just what I was looking for!", "It's great!", "Just like the real VI!". "The documentation is so good I have already learned things about VI that I never knew before." — *IEEE Software*, September 1986.

PC/VI is available for IBM-PC's and generic MS-DOS† systems for only $149. Included are CTAGS and SPLIT utilities, TERMCAP function library, and an IBM-PC specific version which enhances performance by as much as TEN FOLD!

PC/TOOLS™

What makes UNIX so powerful? Sleek, Fast, and **POWERFUL** utilities! UNIX gives the user not dozens, but hundreds of tools. Now the most powerful and popular of these are available for your PC! Each is a complete implementation of the UNIX program. Open up our toolbox and find:

• BANNER	• DIFFH	• PASTE	• SPLIT
• BFS	• DIFF3	• PR	• STRINGS
• CAL	• GREP	• RM	• TAIL
• CHMOD	• HEAD	• SED	• TR
• CUT	• MAKE	• SEE	• TOUCH
• DIFF	• OD	• SORT	• WC

All of these for only $49.00; naturally, extensive documentation is included!

PC/SPELL™

Why settle for a spelling checker which can only compare words against its limited dictionary database when **PC/SPELL** is now available? **PC/SPELL** is a complete implementation of the UNIX spelling checker, renowned for its understanding of the rules of English! **PC/SPELL** determines if a word is correctly spelled by not only checking its database, but also by testing such transformations as pluralization and the addition and deletion of prefixes and suffixes. For only $49.00, **PC/SPELL** is the first and last spelling checker you will ever need!

Buy **PC/VI** and **PC/TOOLS** *now* and get **PC/SPELL** for only $1.00! Site licenses are available. Dealer inquiries invited. MA residents add 5% sales tax. AMEX, MC and Visa accepted without surcharge. Thirty day money back guarantee if not satisfied! Available in 5¼", 3½" and 8" disk formats. For more information call today!

*UNIX is a trademark of AT&T. †MS-DOS is a trademark of Microsoft.

CUSTOM SOFTWARE SYSTEMS
P.O. BOX 678 • NATICK, MA 01760
617•653•2555

CSS

UNIX TOOLS FOR YOUR PC

CIRCLE 14 ON READER SERVICE CARD

should be the boss. All sorts of problems can crop up when you try to get the coordination right in a hierarchy. I just touched on some of them in conjunction with merge points and loops in data flow diagrams.

Most of the everyday issues, however, are really quite pedestrian. They have nothing to do with fancy connectedness. Rather, they arise out of the very modules we write all the time to resolve structure clashes and buffer or unbuffer data.

Last month I asked you to consider the effect on a module of making it an input function, a main function, or an output function. Figure 5 shows a very simple data flow diagram that could force you to confront such effects.

Depending on which of the three bubbles you choose as the boss module, the central module (B) can occur in any of three positions in the resulting hierarchy. Figure 6 shows all three possibilities.

Let's assume that B's role in life is to gather several records from A, assemble them into a combined datum, and deliver a stream of such combined data to C. If B is an input module, then C calls B every time it needs another combined datum. In turn, B repeatedly calls upon A until it has assembled a complete record. B then returns control to C with the combined datum as the function return value.

If B is the main module, it is built around a loop. While more input records can be obtained from A, the main module assembles some of them into a combined datum and calls C to dispose of the combined datum.

Finally, if B is an output module, it is called by A every time A constructs a record. B must maintain enough private memory to hold a combined datum, with notes detailing the state of that combined datum. B adds the newest record to the combined datum, calls C if the combined datum is complete, then updates its private memory so that it knows where to start the next time it is called. B then returns control to A.

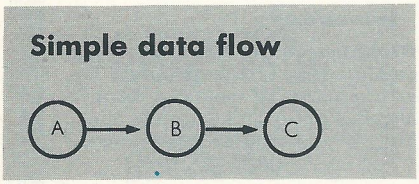


Figure 5.

Of the three implementations, the last is probably the most ornate. You can't avoid having private memory when B is an output module. You should know by now that having private memory incurs a maintenance cost that had better pay for itself by decreasing complexity somewhere in the design.

Any alternate hierarchy that doesn't need private memory should be favored. By this criterion, the implementation that has B as the main function is probably the least complex. (It may lead to higher costs for the other modules, but we'll get to that in a moment.)

The implementation that has B as an input module can be equally simple, provided there are no lookahead issues in recognizing the end of a group of records that constitutes a combined datum.

If you must look ahead, then you need some form of pushback as well. This means either A or B must have private memory to the tune of one input record, plus state information, to assist the input parse. So once again B is the least complex if it is the main function.

You should not be surprised to learn that all modules are least complex when implemented as main functions. That is one of the reasons UNIX-style pipelines are often so cheap to code and maintain. Each module is the boss of its own little data transformation, and the operating system solves most of the coordination problems in a standard way. But if you're determined to implement a data flow diagram as a single hierarchy of modules (a single program), then only one module can be the boss.

That's not as bad as it sounds. As this little example shows, B is almost as simple as an input module. And it is not too bad as an output module. Chances are the other modules in the data flow diagram have different problems. You should be able to guess the properties of a module that unpacks records, for instance, rather than packing them as B does.

An unpacker has private memory as an input module, is simple as a main function, and is equally simple as an output module. None of the implementations are very difficult to write or maintain.

So the seat-of-the-pants rule for picking the best bubble to grab is really quite straightforward: locate the one that has the most complex structure and make it the boss. Chances are that one module stands out from the others in this regard. And the others can probably be implemented reasonably well wherever they fall on the hierarchy.

You will find that this straightforward rule usually selects modules somewhere near the center of a data flow diagram. The fringe modules are usually the easiest to simplify by outside-in design, so they present the fewest problems by the time you get to top-down design.

The approach outlined in this column and my previous column covers several major schools of software design. Each uses a different vocabulary and devotes considerable attention to details I have touched upon fleetingly here. I will spell out the parallels in approaches between schools, and some other details, in my next column.

P.J. Plauger has coauthored several popular textbooks with Brian Kernighan, including The Elements of Programming Style *and* Software Tools in Pascal. *He is secretary of X3J11, the ANSI C standard committee, and convenor of the ISO C standard committee. He is also the president of Whitesmiths Ltd.*

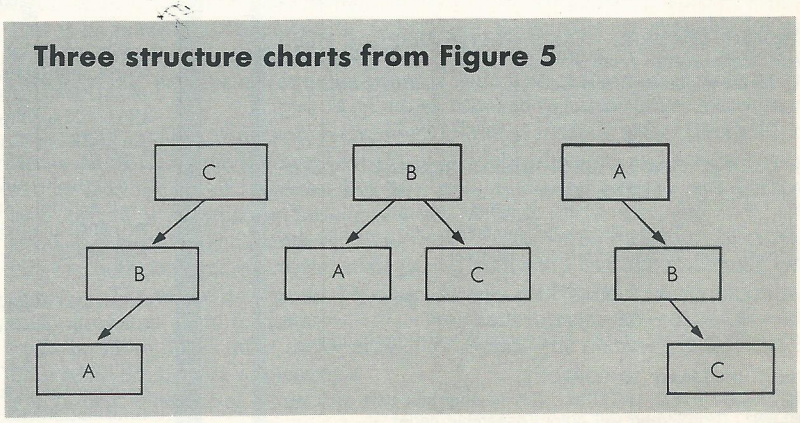


Figure 6.

There used to be a big gap between databases and knowledge bases.

Up until now, if you wanted to have an expert system analyze data stored in a database, you had only two choices:

You could manually enter each piece of data into the expert system knowledge base. Number by number. Letter by letter.

Or you could have programmers skilled in both database and knowledge base programming spend months building a custom database-knowledge base link.

Now you have a better choice.

Introducing KEEconnection.

☐ Tell me more about KEEconnection.™
☐ Keep me informed of upcoming IntelliCorp seminars.

Name

Title

Company

Address

City/State/Zip

Phone

IntelliCorp®

Send to:

IntelliCorp, Inc.
1975 El Camino Real West
Mountain View, CA 94040
415/965-5650

Announcing the grand opening of a new bridge between data and knowledge bases—the KEEconnection™ software from IntelliCorp.

KEEconnection lets you connect knowledge bases developed with IntelliCorp's Knowledge Engineering Environment™ (KEE™) software to SQL® databases in days, instead of months. You can even connect a KEE knowledge base to more than one database. And you can do it all without extensive database programming expertise.

We built an expert into KEEconnection, so you don't have to be one to use it.

A KEE-based expert system built into KEEconnection takes the work out of connecting databases and knowledge bases. You simply use a mouse to "draw" links between database and knowledge base structures pictured on the computer screen.

Then KEEconnection takes over, automatically generating whatever code is needed to move information back and forth. Transparently. Efficiently. Reliably.

Exactly what you'd expect from a well-engineered bridge.

Another great bridge from IntelliCorp.

And KEEconnection is just the sort of product you'd expect from IntelliCorp. Since 1980 we've been providing software and services that bridge the worlds of data processing and "knowledge processing."

Including the first software package to let knowledge system developers build their applications on high-powered workstations, and then deliver them on conventional hardware. Without recoding, and without loss of functionality.

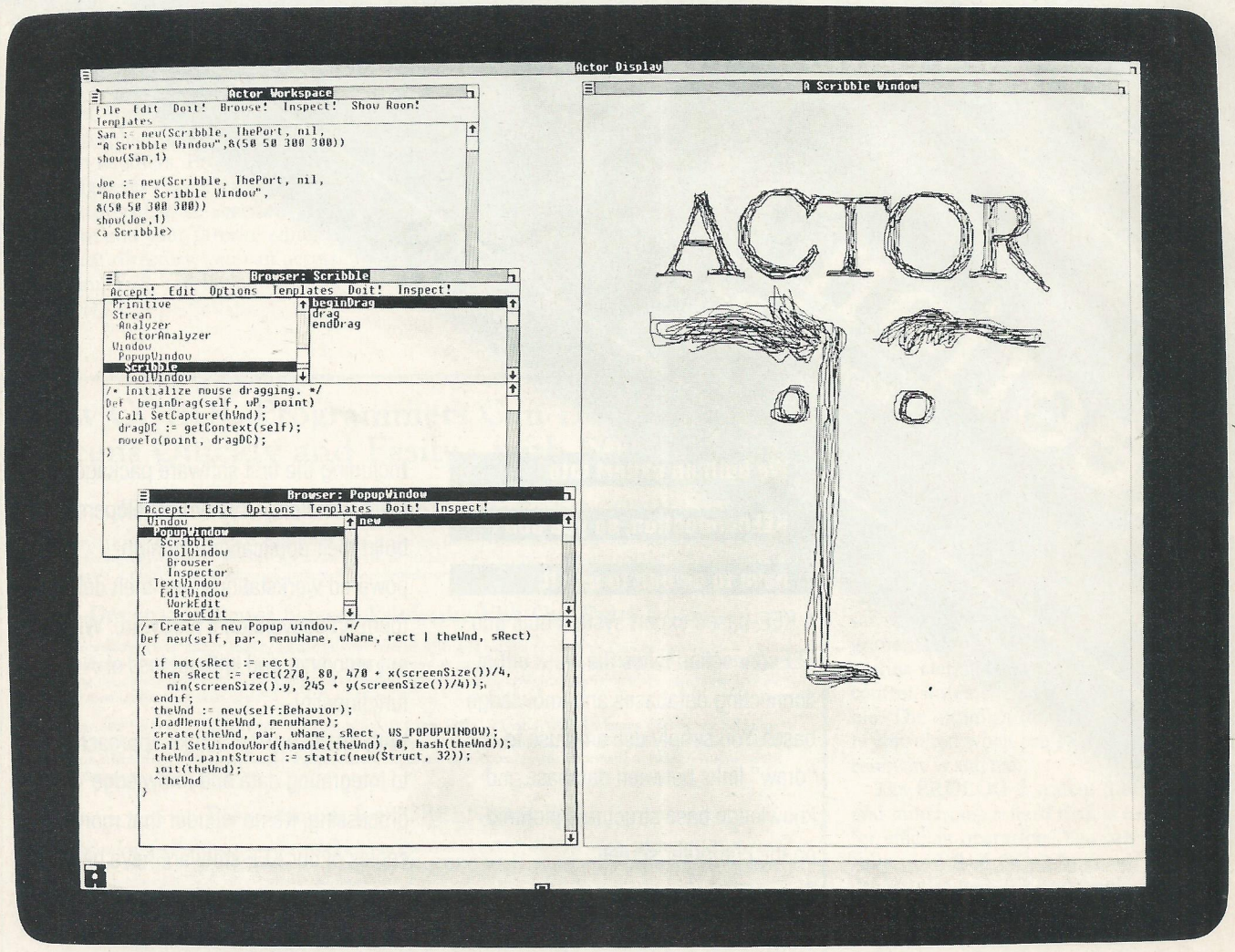
With IntelliCorp's practical approach to integrating data and knowledge processing, it's no wonder that more copies of our KEE software have been licensed than all the other large-scale knowledge system development products—*combined.*

Find out more about KEEconnection, IntelliCorp's new bridge between databases and knowledge bases, by filling out and returning the coupon at left. We'll also keep you informed of upcoming IntelliCorp seminars where you can learn how companies like yours are building their own bridges between the worlds of data and knowledge processing.

KNOWLEDGE ENGINEERING ENVIRONMENT, KEE and KEECONNECTION are trademarks of IntelliCorp, Inc. INTELLICORP is a registered trademark of IntelliCorp, Inc.

CIRCLE 15 ON READER SERVICE CARD

HOW TO WRITE A WINDOWS APPLICATION IN TEN MINUTES.



Actor™ is a new language that combines Microsoft® Windows with object-oriented programming. This means you can produce mouse and window applications very quickly.

For example, we created a simple "paint" program, and used it to draw the Actor logo you see on the screen. The whole program only took ten lines and ten minutes. Part of it is in the middle window on the left.

Above, you see the commands that initialized the paint window and made it appear on the screen. Below, some code that's built into Actor, specifying window behavior. Through a process known as "inheritance," it's called into play automatically.

Try programming in this new way, and you'll never go back.

Find out about Actor. Call The Whitewater Group®, (312) 491-2370.

Technology Innovation Center
906 University Place, Evanston, IL 60201

MICROSOFT® WINDOWS
AN APPLICATION DESIGNED FOR MICROSOFT WINDOWS

CIRCLE 16 ON READER SERVICE CARD

80386 promises a new age for AI

By Susan J. Shepard

The Intel 80386 microprocessor is here—at last. It is proving to be all that we expected, and the last obstacle to 386 supermicrocomputers is an operating system that can avail the developer and user of its power. It will provide a powerful and affordable platform for expert systems and other AI applications that are truly useful.

AI has been been waiting for this chip; many AI applications developers whispering, "When the 386 is available...." and hinting of powerful systems for the desktop and a rich development environment for the programmer working and learning in the real world.

The Intelligent Computer Systems Research Institute at the University of Miami, Coral Gables, Fla., predicts a threefold increase in expert systems this year, largely because the 80386 has arrived. In fact, the processor narrows the gap between micros and LISP workstations significantly—at a substantial dollar difference.

What will micro-based 386 systems offer when an adequate operating system is available for the industry-standard MS-DOS compatibles? What do available 386 systems offer for AI development? Pending a widely available operating system with downward compatibility to prior MS-DOS software, AI tools implemented for that environment will have to run on 386 machines within the constraints of DOS and 16 bits. Is it worth moving up to 386 running under DOS now?

Each of these questions is complex, and many factors must be weighed to ensure that time and money are invested wisely now and prior hardware and software investments are protected. We can start by looking at the 80386 itself and then consider the Inboard 386/AT system, selected for best price, best performance, and compatibility, to see how it handles some basic AI tools for micro development.

We'll also look briefly at Gold Hill's 386-based LISP development system built to deliver an optimized hardware and software environment to existing IBM XT/AT compatibles.

Intel's 80386

Intel's 80386 is a high-performance, 32-bit microprocessor designed for computer-intensive applications such as CAD/CAM, high-resolution graphics, high-level factory and office automation, and publishing. More importantly, it will permit advanced systems generally termed AI to move toward production applications levels. Machine vision, speech recognition systems, and more complex expert systems will become more feasible and affordable. The 32-bit 386 integers and 80-bit 80287/80387 floating point numbers can easily support the speed and accuracy of computation required for fine precision and axis control of an industrial robot.

Frequencies of 12 MHz and 16 MHz are now available, and a 20-25 MHz frequency is forthcoming. The 16 MHz, running without wait states, can achieve a sustained 3 to 4 millions of instructions per second (MIPS). In addition to faster clock speeds, the 16 MHz version has lower clock counts.

This architecture has the programming resources required to directly support applications characterized by large integers, large data structures, and large programs or large numbers of programs. Physical address space is 4 gigabytes; logical address space is 64 terabytes. Eight 32-bit general registers can be used interchangeably as instruction operands and addressing mode variables. It has a complete instruction set for manipulating data types and controlling execution. Data types include 8-, 16-, and 32-bit integers and ordinals, packed and unpacked decimals, pointers, and strings of bits, bytes, words, and double words.

At this speed, memory access is obviously a potentially serious problem. The 80386 bus is designed to use both 45 nanosecond static RAM (SRAM) and slower, less expensive 100ns-120ns dynamic RAM (DRAM) chips. Zero wait state caches are supported, and their size can be anywhere from the useful minimum 4K to the entire physical address space.

Other features are virtual memory support, configurable protection, new debug registers, and object code compatibility. The 80386 has four modes: real, virtual 8086, protected, and native 80386. Real mode is entered after reset; it is essentially a very fast 8086 mode. Virtual 8086 uses special flags, registers, and interrupts to permit fast, safe multitasking in environments that extend the concept of virtual memory in individual programs to include an entire PC.

Protected mode emulates the 80286; at this time neither the 80286 nor 80386 using MS-DOS can take advantage of the protected mode, with the exception of specialized software such as Gold Hill's GCLISP 286 and 386 Developer packages or Texas Instrument's PC Scheme and Personal Consultant PLUS. Native 80386 permits operating systems and programs up to 4 gigabytes of RAM and can address 64 terabytes of virtual memory.

Virtual memory enables the maximum size of a program or programs to be determined by available disk space rather than RAM. AI applications are typically memory intensive, and virtual memory means more flexible delivery of higher performance software on less expensive hardware. Programmers can leave storage management to the operating system rather than writing overlays, to an extent that is not yet clear.

For example, how much real memory is needed to support a certain amount of virtual memory? B.E. Smith and M.T. Johnson, in *Programming the Intel 80386*, estimate that about twice as much virtual memory as real memory can be supported.

The 80386 provides a range of protection facilities that can be applied selectively as the application warrants: separation of task address spaces, from 0 to 4 privilege levels, typed segments, access rights for segments and pages, and segment limit checking. Protection checks are performed in the on-chip pipeline to maximize performance. These facilities are particularly useful when developing programs the size the 80386 is

built to handle—hundreds or thousands of modules.

Four on-chip debug registers enable code or data break points. The registers are independent of the protection system and can be used by all applications. Conventional debugging through a break point instruction and single stepping are supported as well.

Object-code-level compatibility with both the 80286 and 8086/88 processors protects existing software investments and allows rapid development for the 80386. 80286 and 80386 programs can run concurrently. The 80286's architecture is a proper subset of the 80386's, so existing 80286 operating systems and applications can be ported without change.

Virtual 86 is a compatibility feature that establishes a protected 8086/88 environment within the 80386's protected multitasking environment. 80386 paging can give each Virtual 86 task a 1 MB address space anywhere in the physical address space and an 8086 environment. It is possible to build systems that could execute 80386, 80286, and 8086/88 software concurrently. 8086 software can run in the real mode of the 80386, but timing-dependent programs may need modification to function properly in the much faster mode.

Hands on

The Intel Inboard 386/AT add-in 80386 system with 80287 or 80387 for the IBM AT and close compatibles offers a reasonably priced, high-performance entry to 80386-based micros. We will look closely at that product, and run some AI applications on it (Tables 1-4), as a way

Benchmarks for PC Scheme (sec)

Test	80286 640K	80286 2 MB	80386 640K	80386 2 MB
Prime Sieve	16.70	30.71	8.85	15.49
Fib(60) (iterative)	.06	.06	−.01	−.01
Fib(60) (floating point)	.38	.44	.11	.11
TAK	16.02	35.59	8.12	9.0
DERIV	81.40	134.51	48.71	78.98
DDERIV	103.21	167.58	51.85	100.95
DIV2 (iterative)	21.42	34.16	9.06	16.04
DIV2 (recursive)	62.62	109.57	27.35	40.75
Factorial (100)	.11	.17	.11	.11

Results are for PC Scheme on 80286/8MHz and 80386/16 MHz.
Conventional (640K) and extended memory (2MB) mode Scheme used for each processor.
No software or hardware was changed except for Inboard 386/AT.
No math coprocessor installed in either configuration.

Table 1.

GCLISP benchmarks (sec)

Test	80286	80386
TAK	57.39	32.29
DIV2 (iterative)	78.05	56.95
DIV2 (recursive)	98.30	67.39
DERIV	111.43	58.38
DDERIV	129.19	100.03

Results are for GCLISP v. 1.01 interpreter.
Tests run in conventional (640K) memory for each configuration. GCLISP does not support bignums.

Table 2.

Now with extended/expanded memory support

PC Scheme—
a simple, modern LISP
for under $100.

Texas Instruments presents PC Scheme, the $95* solution to your symbolic processing needs.

Whether you're an experienced LISP user, or just beginning to discover the power of symbolic programming languages, PC Scheme is the right product for you. It runs on IBM® Personal Computers, as well as the TI Professional Computer family, including the Business-Pro™ computer.

Powerful features include an optimizing incremental compiler for ease of programming and fast execution; an EMACS-like editor; extensions for debugging, graphics, and windowing; DOS-CALL capability; and a programming system for the development of object-oriented applications—all designed to work efficiently on personal computers.

To order, or for more information, call toll-free:

1-800-527-3500

*Suggested list price.
Business-Pro is a trademark of Texas Instruments Incorporated.
IBM is a registered trademark of International Business Machines Corporation.

TEXAS INSTRUMENTS

Creating useful products and services for you.

© 1986 TI 261765-02A

A Multiwindow Screen Editor for Programmers

XTC is more than just a powerful programmer's editor. Its implementation actually encourages users to develop software systems in a more efficient way.

XTC was originally written for a Compupro 8086 machine running CP/M-86, for use in robotic software development. It was called GLACIER, an acronym for "Good Lord, A Centralized Interactive Editing Routine". GLACIER had multiple windows and the ability to share text between windows, and a set of editing commands which were based on the most popular word processor at that time —Wordstar. Windows were important because it was necessary to view several different text files or several different parts of the same text file without constantly switching back and forth between them. Originally written in Pascal MT+86, GLACIER was rewritten for the IBM PC family in IBM Pascal. Blocks, text buffers, and a macro programming language were added to the new editor, and it was released as XTC version 1.2.

Now, nearly three years later, XTC 3.0 is a complete text editing environment with over 100 commands in seven categories: File Interface, Windows, Blocks, Text Processing, Editor Programming, Multitasking, and Special Editing. It comes with its source code on disk, and a full-size, laser-typeset, 180+ page manual. Unlike other programmer's editors, XTC is user-modifiable at the source code level, so you can add to XTC's basic command set, or change how the commands work entirely.

How XTC works

The editor's design was based on its ability to share text across windows and buffers (which are just windows that don't get displayed). Pascal's strong typing lead to the storage of text in a doubly-linked list of line buffers (hereafter called a "*text stream*"), so that cursor movement commands could move a "current pointer" around the text with simple pointer assignment operations. This also makes insertion of text very efficient, since it can be accomplished with one short memory move to make room for a new character.

XTC keeps track of a doubly-linked list of data structures called *window descriptors* which describe the windows displayed on the screen. There is also another doubly-linked list of window structures which are not displayed by the display algorithm for XTC's buffers. Each window descriptor contains information about the physical coordinates of the window on the screen, how big the window is, a pointer to a text stream where the window is positioned over, and a pointer to the current cursor position within the window. Other information, such as the current filename and options like wordwrap and autoindent are also saved in the window descriptor. Figure 1 shows how XTC's principal data structures are connected.

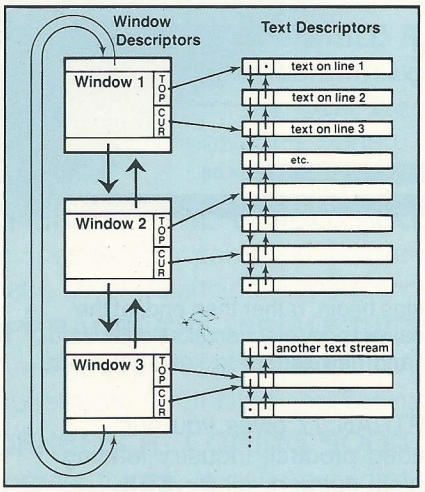


Figure 1. XTC's text and window management data structures. Here two windows share a text stream, and a third is positioned over a different stream.

How to use XTC

Knowing about how a program works makes it possible for you to use it in the best way possible. Obviously, XTC's data structures make for very fast zooming through a text stream, and insertion. It also makes it easy to make new windows quickly and either share text among them or look at other files while you concentrate on the main task of creative development.

There is no free lunch, unless you learn the system.

It takes XTC longer to read a file into data structures, and allocate memory for data structures on the fly for each line descriptor than it does for an unstructured editor to read text into big block buffers. Therefore, programs developed under XTC tend to be composed of a main file and many *include* files. XTC's main draw then, is its windowing system which makes you actually want to put a separate C function in a different *include* file. By storing each function in a separate file, XTC lets the operating system manage your library of source, and it becomes easy to share source routines among several software products.

This system works very well. We've used it in the design of our latest operating system, Wendin-DOS. The new operating system is composed of over 32 modules written in C and assembly language, and each module source file (named *modulename.C*) uses *include* statements to draw in header files that describe data structures, and function files (named *functionname.I*). Altogether, XTC and the operating system manage over 50,000 lines of source code in 500 different source files. The simple convention of using a function's name for its source file makes it easy to retrieve the exact piece of code you're interested in, without wading through tons of code in a single big source file.

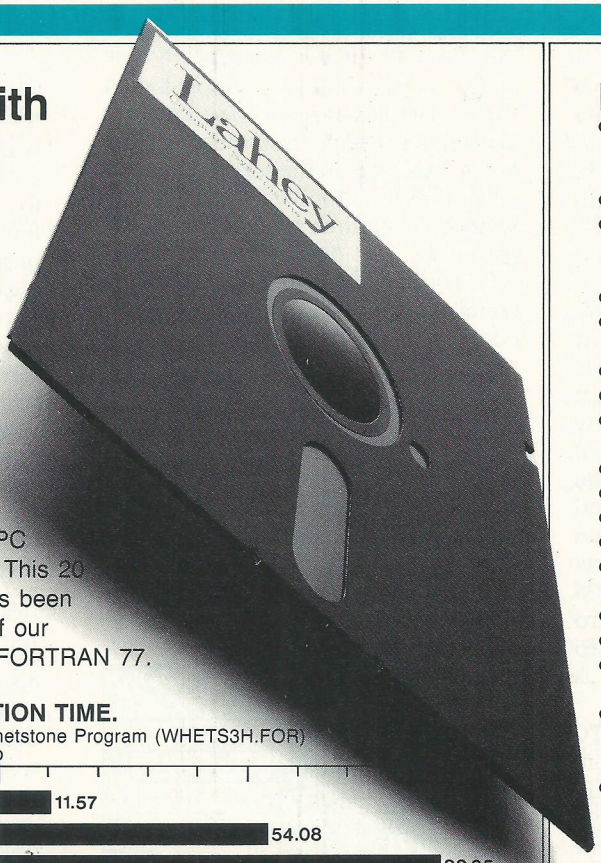
Next month, we'll present XTC's programming commands and macro languages. If you don't already have XTC, get your copy and follow along. You simply won't get the chance to do that with any other editor.

XTC: $99
Wendin, Inc.
P.O. Box 3888
Spokane, WA 99220
(509) 624-8088

Lahey Computer Systems, Inc. Sets a New FORTRAN Standard!

Introducing the latest addition to our line of FORTRAN Language Systems—
Lahey Personal FORTRAN 77.

What you Get With Lahey Personal FORTRAN 77:



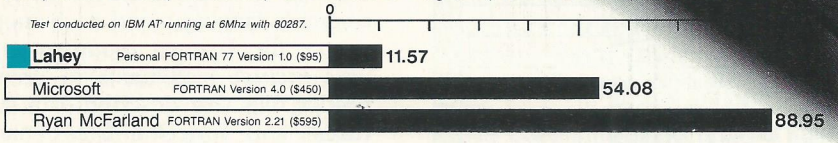
Lahey Experience.

We are experts in designing and implementing FORTRAN Language Systems. Lahey has been producing mainframe implementations since 1967 and a PC FORTRAN (F77L) since 1984. In fact, F77L was named the "EDITOR'S CHOICE" among PC FORTRANs by *PC Magazine*. This 20 year span of specialization has been incorporated into the design of our revolutionary Lahey Personal FORTRAN 77.

LAHEY SLASHES COMPILATION TIME.

Compilation times (in seconds) for Whetstone Program (WHETS3H.FOR)

Test conducted on IBM AT running at 6Mhz with 80287.



Customer Support:

Our philosophy is that customer relationships begin, rather than end, at the point of sale. Services include free technical support, electronic bulletin board for fast service and information access, and newsletters to keep you up to date.

Purchasing the Lahey Personal FORTRAN 77 gives you our FORTRAN experience, a feature loaded product, industry leading compilation speed, and quality technical support; all for **$95.**

International Representatives: Canada: Barry Mooney & Assoc., Tel. (902)6652941 • England: Grey Matter Ltd., Tel. (0364)53499 • Switzerland: DST Comp. Services, Tel. (022)989188 • Denmark: Ravenholm Computing, Tel. (02)887249 • Australia: Comp. Transitions, Tel. (03)5372786 • Japan: Microsoftware Inc., Tel. (03)813822

MS-DOS & MS FORTRAN are trademarks of Microsoft Corporation.

Feature Loaded:

- Full implementation of the ANSI X3.9-1978 FORTRAN Standard
- Fast Compilation (see chart)
- Popular Language Extensions highlighted in the manual
- Source On-Line Debugger
- English Diagnostics and Warning Messages
- LOGICAL*1, LOGICAL*4
- INTEGER*2, INTEGER*4
- REAL*4, REAL*8, and DOUBLE PRECISION
- COMPLEX*8, COMPLEX*16
- Recursion
- 31-Character Names
- Trailing Comment
- Cross Reference and Source Listings
- 64 KB Generated Code
- 64 KB Stack Storage
- 64 KB Commons, Constants and Saved Local Data
- Math coprocessor requirement gives maximum performance
- 350 Page User Manual

SYSTEM REQUIREMENTS:
256K Ram MS-DOS (2.0 or later)
Math Coprocessor Chip (8087 or 80287)

$95

Lahey is setting the PC FORTRAN Standard.

TO ORDER

1-800-548-4778

Lahey Computer Systems, Inc.
P.O. Box 6091
Incline Village, NV 89450
Telephone: (702) 831-2500
TELEX: 9102401256

If you want to program in FORTRAN - Lahey is the Source

☐ Please send me ___ Lahey Personal FORTRAN 77 copies @ $95.00 each
Disk Format ___ 5¼" ___ 3½" _____
Add $2.50 for shipping and handling per unit _____
Nevada residents add applicable tax _____
TOTAL ENCLOSED _____

SEND ME INFORMATION:
☐ Lahey Personal FORTRAN 77
☐ A future release not requiring math coprocessor
☐ LCS Software Products

ENCLOSED
___ Check ___ Visa ___ Master Card ___ AMEX
Number _____
Expiration Date _____
Signature _____
Name _____
Company _____
Address _____
City _____
State _____ Zip _____
Phone _____
Send to: Lahey Computer Systems,
POB 6091, Incline Village, NV 89450-6091

Lahey
Computer Systems Inc.

CIRCLE 18 ON READER SERVICE CARD

It is possible to build systems that could execute 80386, 80286, and 8086/88 software concurrently

of indicating the tremendous improvement in an AI development project using the 80386.

Other products giving good performance are, of course, the Compaq 386 series of personal computers and the Gold Hill 386 LISP System. The latter is an add-in board for the IBM PC XT or AT with a sophisticated LISP development software package.

Because these systems are based on the 80386, my earlier discussion applies to each of them, with the exception of SRAM caching and 80387 support. Cache implementations are very important for achieving good performance, and different approaches should be evaluated before purchase. Vendors should be well-queried about performance and compatibility for your needs.

The marketplace for 80386-based AI is changing rapidly. Joining Gold Hill in the race to exploit the chip's power—with or without an operating system—other major vendors are making announcements. Lucid Corp., which offers AI tools for workstations, IBM PC RT, and other large systems, is developing a package that will run under UNIX on the 80386. Because UNIX is widely used on workstations, ports will increase for both the operating system and product.

My evaluation of the options led me to choose the Inboard as the best solution for my needs, offering compatibility both with present software and future expan-

Various performance considerations (sec)

Task	80286/8MHz	80386/16MHz
Norton SI	7.7	18.7
Load:		
GCLISP 1.01	16.53	6.47
PC Scheme ext.	10.49	5.22
PC Scheme conv.	8.20	4.13
Personal Consultant PLUS ext.	26.40	15.31
PC+ 360K KB	15.21	7.43
Kedit and 16K file	5.99	3.10

Table 3.

Inboard 80386/AT with 80387 coprocessor (sec)

System/ using	Whetstone[1]	Dhrystone[2]	TAK[3]	DDERIV[3]	IDIV2[3]	FIB(60) (f-p)
Inboard	395,000	2,083	8.13	42.40	9.01	.09
Compaq386	175,000	3,333	N/A	N/A	N/A	N/A
SUN-3/T language	N/A	N/A	0.25	4.92	0.24	N/A
SUN 2/120	N/A	806	N/A	N/A	N/A	N/A
80286 3000HL	N/A	1,041	16.02	103.21	21.42	.38

1. Curnow, H.J. and B.A. Wichmann: *Computer Journal.* vol. 19, no. 1.
2. Weicker, R.P.: *Communications of the ACM.* vol. 27, no. 1.
3. Gabriel, R.P.: *Performance and Evaluation of LISP Systems.* MIT Press, 1985.
N/A = Not available.

Inboard 80386/AT runs at 16 MHz. 80387 coprocessor runs at 16 MHz. MS-DOS 3.2. Gabriel benchmarks run with Texas Instruments PC Scheme v. 2.0 in conventional memory mode. Performance of other systems offered for general comparison only.

Table 4.

Dbase*

programming tools
*Clipper, FoxBASE+,
dBASE, QuickSilver

The UI Programmer

UI is the first professional code generator; we wrote UI for programmers who want to automate their work but cannot use code that is 'almost' good enough. If your user interfaces include bounce-bar menus, pop-up help screens and the other features of today's best programs, you will gain an order of magnitude in productivity with UI.

UI is a second generation, programmable product — so your code comes out your way. Application specific edits, for instance, can be placed in the UI 'template' which controls the generation. Edit the screen appearance until it 'looks and feels' perfect. Everytime you generate code, your special logic is preserved.

Speaking of editing the screen, UI includes a powerful, 3-D screen editor, so you can draw pop-up help boxes over your pull-down menus, over your application.

The Documentor

To run Doc, you just tell it the name of the main-line routine and make sure your printer has a lot of paper! (Sure, you can have the output go to the screen or a file, too.)

You can tailor your documentation to include any or all of: a table of contents, system tree diagram (main line is the root), hierarchy (box diagram) charts for each module, action diagrams (modern style flow charts) for each PRG or procedure, DBF listings (structure, indexes, more), where used/updated listings for fields and all variables — by module and by line number within each module.

Our written money-back satisfaction guarantee set a new standard when we began it in 1985. (Return rate to date: 9.6% and dropping!) No copy protection, royalties or other nonsense.

Suggested retail: $295 each, (800) support included. At your dealer today. Call us for a very special offer on our latest release! (800) 233-3569 or, in NY, (212) 406-7026.

WallSoft
The Computer Aided Software Engineering Corporation
233 Broadway, Suite 869, New York, NY 10279

CIRCLE 19 ON READER SERVICE CARD

sions. Its performance for the languages and tools I use, and its overall system improvement, has exceeded Intel's claims for the board. I found that I not only realized a two- to three-fold increase in execution times for software at 16 MHz, but I also got better performance indexes from a rather old Seagate ST-225 hard disk using the CORE disk performance test v. 2.7. This was a pleasant surprise, but replacing the disk with a faster unit would improve performance tremendously.

Specifications for the board are: speed selection ranging from 16 MHz to the original system speed; 256K x 32-bits memory size; 120ns DRAMs up to 3 MB of 32-bit memory and access to other extended or expanded memory in the host system; and a direct-mapped 45 nanosecond SRAM cache, 64K x 32 bits. Intel claims that 90% of the 80386's memory requests are filled from this 64K cache instead of the older, slower system memory.

Overall performance indicates that hits are at least 90% when memory outside the zero wait-state cache must be accessed and wait states are introduced. However, since these are lower clock count 80386 waits, the effect is not as severe as on the 80286.

The board uses a 16-bit slot in the host machine. The 80286 chip is replaced by an ingenious cabling technique from the system board CPU socket to the 80386 board. Control software bridges differences in 80386 and 80286 error handling and offers other easy-to-manage drivers. Speed is software-switched.

The move from 8 MHz to 16 MHz has caused no compatibility problems with any of the current AI language implementations or expert system tools for MS-DOS, with the exception of the Gold Hill GCLISP 286 Developer. This product will not run on the Inboard, but the 386 Developer will run on the Compaq 386 and Gold Hill's Hummingboard.

Compatibility with the Inboard could not be tested. Because the performance of these individual applications varies widely, the impact of the 80386 can best be described as a big improvement. Waits for notoriously processor-intensive LISP interpreters are reduced anywhere from 25% to 75%. Shells improve in a similar fashion.

The benchmarks for extended memory PC Scheme (Table 1) make that package's 2 MB memory capability attractive. Gold Hill's GCLISP interpreter (Table 2) improves significantly as well.

The improvement in 80286 or 8086/88 AI tools is significant without even attempting to optimize code for the environment. The 80386 architecture offers a great opportunity for AI. Very soon, AI tools that can exploit this architecture will join Gold Hill's package, and applied AI may finally emerge as a major element in practical computing.

References

Abelson, H., and G.J. Sussman. *Structure and Interpretation of Computer Programs.* The MIT Press, 1985.

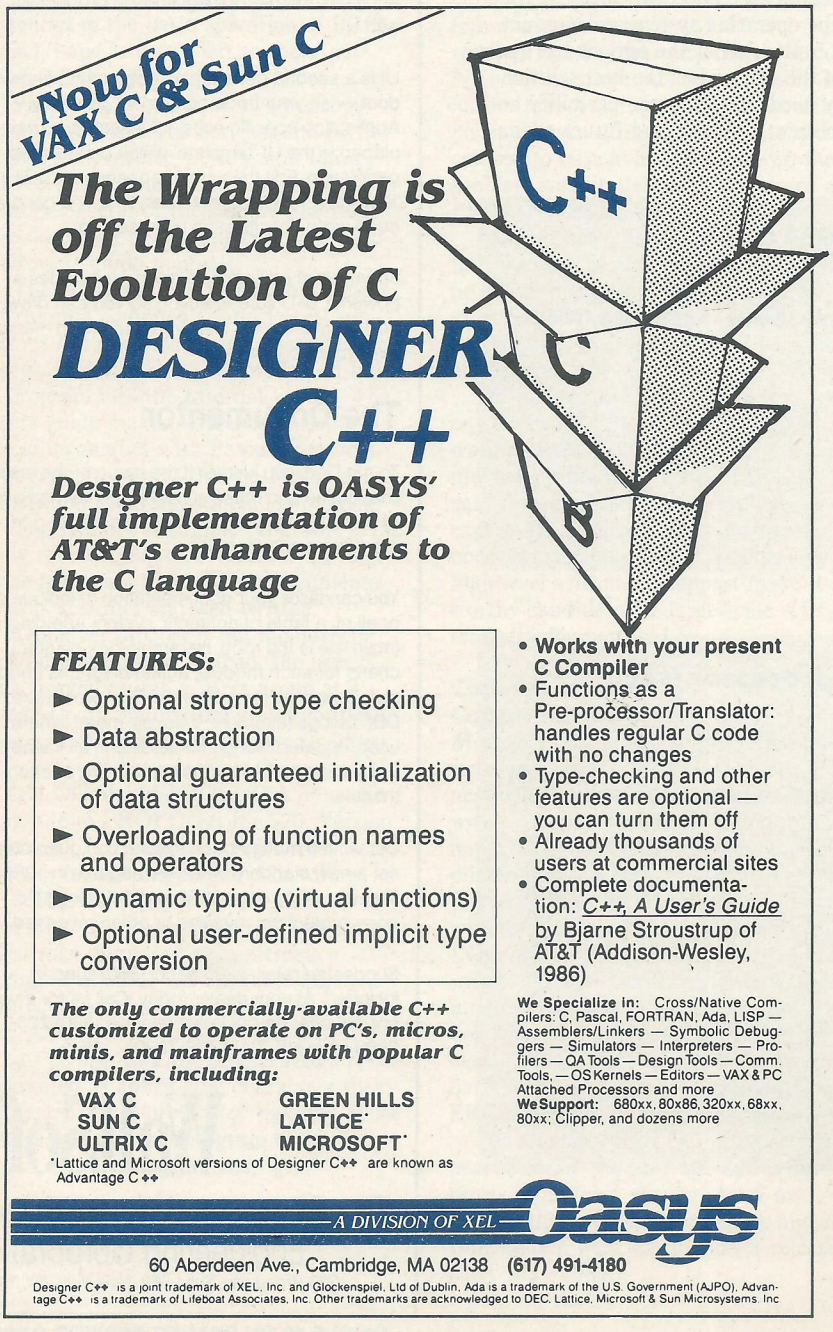
Gabriel, R.P. *Performance and Evaluation of LISP Systems.* The MIT Press, 1985.

Introduction to the 80386. Intel Corp., 1986.

Smith, B.E., and M.T. Johnson. *Programming the Intel 80386.* Scott, Foresman and Co., 1987.

Source code for the benchmarks is available on the COMPUTER LANGUAGE *Bulletin Board Service and CompuServe forum.*

Susan Shepard is a writer and editor specializing in the microcomputer field. She is the AI Insider columnist for AI EXPERT. *In addition to doing private consulting, she is a program consultant for the Academy for Educational Development, an international consulting firm based in New York, N.Y.*



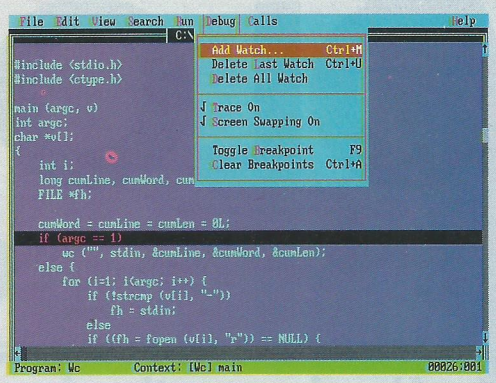
CIRCLE 20 ON READER SERVICE CARD

MICROSOFT LANGUAGES NEWSLETTER VOL. 2, NO. 7

News about the Microsoft Language Family

New Microsoft® QuickC™ Compiler Provides Easy C Programming Through Total Integration

Many of you have been waiting for a $99 C compiler designed to help get your programs running quickly. Microsoft QuickC Compiler Version 1.0, being released in the third quarter of 1987, comes with a completely integrated development environment allowing you to edit, compile and debug your programs without ever leaving QuickC.



Microsoft QuickC at Work.

The key features of QuickC include an in-memory compiler, an integrated source-level debugger, an in-memory MAKE utility and a full-screen editor. A supplied graphics library with a wide range of screen-control functions allows you to take advantage of the extensive graphics capabilities of the IBM® Personal Computer. A stand-alone MAKE utility, completely compatible with the in-memory MAKE, is provided. QuickC includes a LIB utility for creating, organizing and maintaining object module libraries and a LINK utility for combining relocatable object modules into an executable program.

Write Bug-free C Programs More Easily with the Microsoft QuickC Integrated Debugger

Microsoft QuickC has an integrated source-level debugger that lets you see exactly what your program is doing. This makes writing bug-free C programs easier than ever before. Borrowing from the CodeView™ debugger technology in Microsoft C, the QuickC debugger lets you pinpoint errors by stepping through the source code while it executes, using animate, trace, or single-step mode. Set, examine and clear dynamic breakpoints to stop execution as needed so you can take a closer look at what your program is doing. And find out how you got to a particular point in your program by backtracing within the stack to check past history. The Watch Window lets you observe the contents of both local and global variables and see them change as your program executes. Use the screen-swapping feature for screen-intensive applications, switching between the source code and program output as you debug. Best of all, when the bug is fixed just hit the ‹F5› key and your program compiles and runs until the first breakpoint is reached.

In-Memory Compiling and Editing Speed Up Programming

In-memory compilation makes Microsoft QuickC extremely fast: On an IBM PC AT, it compiles and links at 7,000 lines per minute. It can catch up to 26 errors during a single compilation, allowing you to fix all problems before recompiling. During recompilations, the in-memory MAKE utility saves you additional time by creating the MAKE file for you and recompiling only the changed modules.

At the end of a compilation, the editor helps you by placing the cursor at the point of the first error and subsequent errors. It also speeds the correction process by providing detailed information about the nature of each error.

The compiler's integrated full-screen editor includes a broad range of helpful functions, including Cut, Copy, Paste, Undo, Search and Replace, Overtype, and Insert. A "hot key" allows you to toggle between two program modules when editing multiple-module programs. And there's context-sensitive, on-line help to answer your C language and library questions.

For more information on the products and features discussed in the Newsletter, **write to: Microsoft** Languages Newsletter 16011 NE 36th Way, Box 97017, Redmond, WA 98073-9717. **Or phone:** (800) 426-9400. In Washington State and Alaska, call (206) 882-8088. In Canada, call (416) 673-7638.

Latest DOS Versions:	
Microsoft C Compiler	4.00
Microsoft COBOL	2.20
Microsoft FORTRAN	4.00
Microsoft Macro Assembler	4.00
Microsoft Pascal	3.32
Microsoft QuickBASIC	3.00

Microsoft and the Microsoft logo are registered trademarks and QuickC and CodeView are trademarks of Microsoft Corporation. IBM is a registered trademark of International Business Machines Corporation.

PROLOG and C

AI and conventional languages don't have to be at odds—they can complement each other's strengths

By M. Haytham Matthews

The recent proliferation of computer languages presents an interesting challenge to software developers. Given that each language—or, more precisely, each implementation—has its own share of strengths and shortcomings, making the best choice can be something of a dilemma.

Information Resource Management Associates was recently involved in an effort to take an application developed on a LISP machine and reimplement it on a PC/AT. The experience gave rise to issues germane to the selection of an implementation language as well as to the nature of multilanguage programming in PROLOG and C.

The application in question required some data base functionality (provisions for adding, editing, and deleting data) as well as an inferencing component. The inferencing had to be done in real time, and the data had to be easily accessible and sorted in different but predictable ways.

In addition, the interface could not be an exact replica of the application interface on the LISP machine because of differing screen characteristics (smaller dimensions, etc., on the PC/AT). But it had to be clear, consistent, and very easy to use. Furthermore, since changes to the design of the application appeared likely en route to the delivery system, flexibility was at a premium.

Choosing a base language

The criteria used to choose an implementation language were based on the assumption that the PC/AT prototype would serve as a feasibility analysis and would later need to be rewritten in a suitable delivery language. (The prototype has since been judged sufficiently robust to move directly into the delivery environment on a trial basis.)

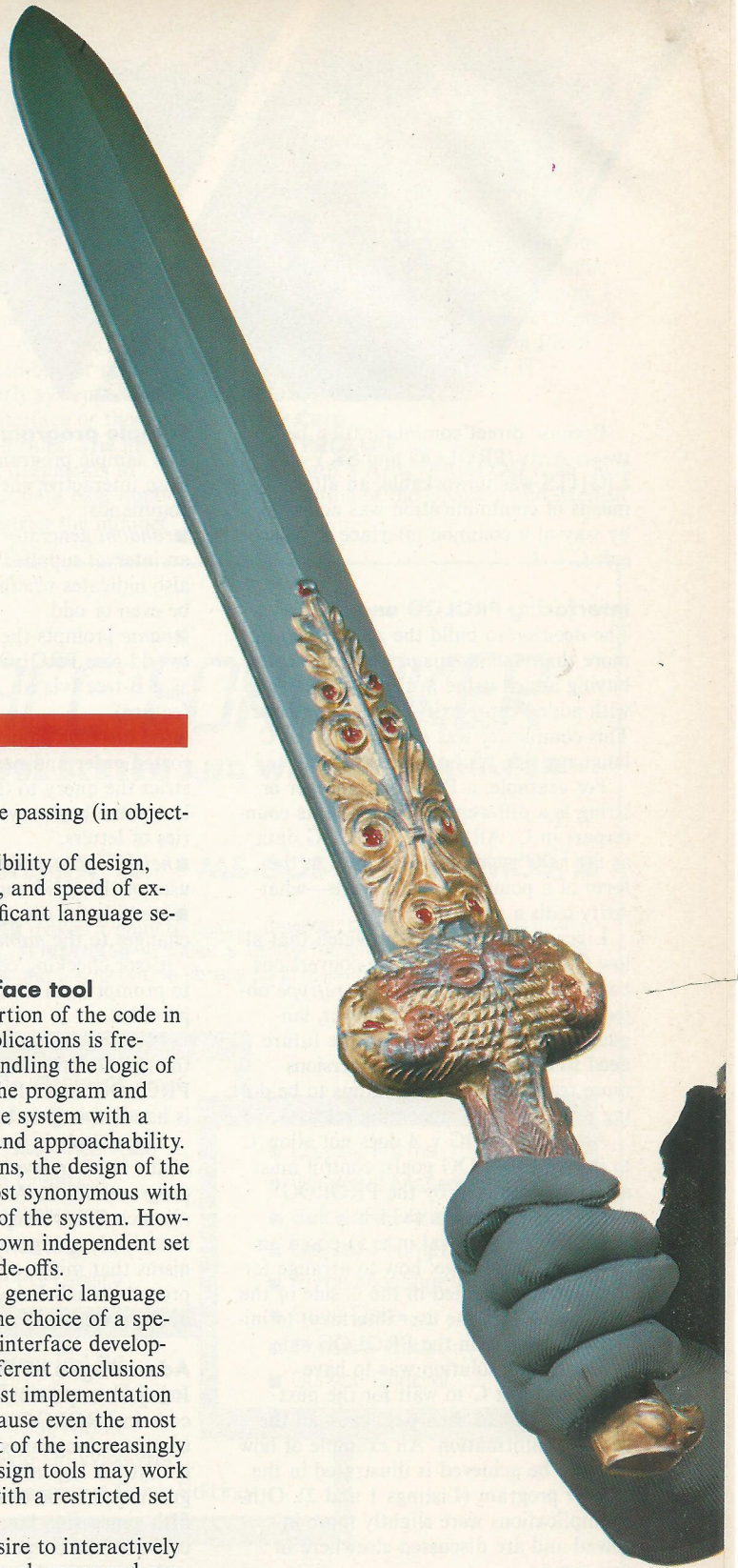
The ideal language on which to base such a prototype would have been the same one that supported the program on the LISP machine. As that language was unavailable for PC/AT-class machines, PROLOG was selected as an appropriate alternative.

PROLOG is a fifth generation language that is more widely known and arguably more stable in its structure than many of the artificial intelligence languages and tools on the market. In the terminology of expert systems, it is a backward-chaining symbolic language with a built-in, depth-first search control regime based on unification and Horn logic.

PROLOG is relational in the sense of relational data base languages such as IBM's SQL. However, it is much more than a data base language, having been used extensively for AI and expert systems applications.

Nevertheless, integrated data base features figured highly in our decision to use PROLOG. The extent of data base functionality is one area that distinguishes one PROLOG implementation from another.

Join Forces



In our case, Arity Corp.'s Arity/PROLOG v. 4 was a good choice, partly because of its built-in facilities for handling B-trees and hash tables. Another factor was the availability of a compiler for Arity/PROLOG (PROLOG is commonly an interpreted language). Thus if the language was as well suited to the problem as we thought, we would be giving the possibility of real-time inferencing a reasonable shot.

Another option was to use a conventional language like Pascal or C. Even with a compiler, fifth generation languages running on standard hardware generally cannot rival more conventional ones in execution speed. But this is partly a consequence of a trade-off between power of expression and run-time performance.

Fifth generation languages are valuable in large measure because they help reduce the cognitive distance between the verbal description of a process or solution and its representation in executable code. The concepts and logic that constitute the design of a program are expressed more directly in such a language and are less burdened by nonmeaningful declarations and restrictive syntactic requirements. The resulting programs are thus easier to both read and change. However, this clarity and malleability is paid for at run time.

Among the features that help increase the cogency of fifth generation language programs, at the expense of execution speed, are the dynamic binding of variables and types, automatic memory management, and message passing (in object-oriented languages).

In our project, flexibility of design, speed of development, and speed of execution were all significant language selection criteria.

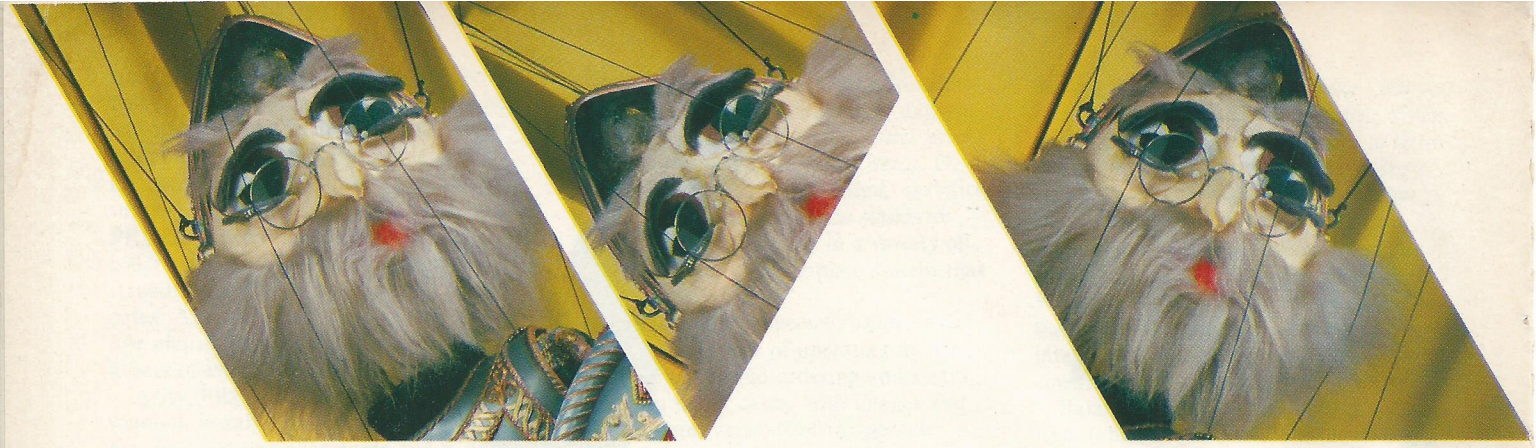
Choosing an interface tool

A considerable proportion of the code in end-user-oriented applications is frequently devoted to handling the logic of interaction between the program and user and endowing the system with a sense of smoothness and approachability.

In many applications, the design of the user interface is almost synonymous with the high-level design of the system. However, it also poses its own independent set of challenges and trade-offs.

Applying the same generic language selection criteria to the choice of a specialized tool for user interface development may lead to different conclusions about which is the best implementation language. This is because even the most language-independent of the increasingly powerful interface design tools may work (or work well) only with a restricted set of languages.

In our case, the desire to interactively design screens with touch zones and pop-up windows leads to the selection of a product called SKYLIGHTS, produced by Skylights Software. (For a review of this product, see "Program prototypers and the illusion of reality," *COMPUTER LANGUAGE,* Jun. 1987, pp. 109-112.)



Because direct communication between Arity/PROLOG and SKYLIGHTS was unworkable, an alternative means of communication was achieved by way of a common interface to Microsoft C v. 3.

Interfacing PROLOG and C

The decision to build the application in more than one language was, in effect, buying added value in the user interface with added complexity in program code. This complexity was felt most in the C language side of the application.

For example, a PROLOG integer or string is a different animal than its counterpart in C. All Arity/PROLOG data, as far as C is concerned, comes in the form of a pointer to a structure—what Arity calls a *reftype*.

Library functions are provided that allow C code to perform type conversions back and forth between these *reftype* objects and C data types. However, language interface designers of the future need to make these type conversions more transparent (Arity claims to be doing just that in its upcoming release).

Arity/PROLOG v. 4 does not allow C to invoke PROLOG goals; control must always be retained by the PROLOG code. This restriction (which is also scheduled for removal in v. 5) posed an interesting challenge: how to arrange for user actions detected in the C side of the application (via the user interface) to initiate processing in the PROLOG side.

The simple solution was to have PROLOG ask C to wait for the next user request and then pass back all the relevant information. An example of how this can be achieved is illustrated in the sample program (Listings 1 and 2). Other implications were slightly more involved and are discussed elsewhere in this article.

One stylistic point that deserves special mention is the use of dispatch procedures and functions to encapsulate the communication channel between the two languages. This approach, along with mnemonic and other conventions, pays off in debugging and maintenance.

Sample program description

The sample program in Listings 1 and 2 is an interactive shell that accepts five commands:

- *random* generates a random number in an interval supplied by the user. The user also indicates whether the number should be even or odd.
- *name* prompts the user for a new name to add to a PROLOG data base stored as a B-tree (via an Arity built-in feature).
- *list* displays names in the data base in sorted order and permits the user to restrict the query to those names where the last name begins with a user-supplied series of letters.
- *help* displays a message informing the user of the valid commands.
- *bye* bids the user good-bye and saves all changes to the *nametree* data base.

Error checking consists of continuing to prompt until a valid response is provided.

Most of the logic and functionality in the sample program is handled by the PROLOG code. The user interface I/O is handled largely by the C code.

The sample is meant to illustrate some of the issues relevant to two-language programming in Arity/PROLOG and C. As such, I have kept away from using complex language constructs and mechanisms that might otherwise have improved the efficiency or generalizability of the code.

Advantages of PROLOG

It is just as possible to write inscrutable code in PROLOG as it is in most other computer languages. My claim, however, is that the logical foundations of a program tend to be closer to the surface in fifth generation languages than in more conventional ones.

As an example, even though the reader may not have extensive knowledge of the PROLOG language, many of the basic ideas and definitions that make up the sample PROLOG program (Listing 1) can be surmised with a bit of persistent study.

The smooth integration of data base functionality with the PROLOG code is another important feature of PROLOG implementations like Arity's. This integration can be very useful in prototyping decision support systems as well as creating stand-alone applications that require flexible but limited data base support. (PROLOG implementations by themselves have yet to support data base functionality at the same level as separate data base products.)

The sample program contains an example of the use of a B-tree for storing full names sorted by last name followed by first and middle names.

An interesting aspect of the PROLOG language is the ease with which additional features can be added to procedures through the addition of rules. The procedure *inbetween* in the sample program, for example, is able to serve a dual purpose. If the first argument is an instantiated variable (if it has a value), it will perform a test to determine if that value lies in the interval defined by the second two arguments. Alternatively, if the first argument does not contain a value, it will be instantiated to a random number within the interval bound by the second two arguments.

Many features of the PROLOG language make it a potentially useful tool for prototyping. It is also useful for application development in situations where flexibility is more important than speed.

However, much depends on the quality of the PROLOG implementation and, in particular, on the extent of the built-in features analogous to library functions in languages such as C.

Advantages of C

The C language is distinguished by speed and conciseness as well as flexibility. But whereas PROLOG supports conceptual flexibility in the sense of readability and ease of modification, C supports flexibility in the sense of getting the most out of the machine.

This kind of flexibility and speed has made C a good language for developing user interface tools that directly manipulate the video buffer, etc. Interfacing to C means gaining access to such tools as well as to the extensive library functions developed for C.

The sample program in Listing 2 pro-

vides two examples of C functionality that would have been unavailable had the program been written exclusively in PROLOG.

The *printf* and *scanf* statements present in all commercial C compilers provide for several I/O formatting features generally unavailable in the basic PROLOG product. This functionality is used to format the output generated by the *list* command in the sample program.

The random number generator provided by Microsoft C was also used instead of the one available in Arity/PROLOG because an integer value was desired. Arity's generator returns a real number between 0 and 1.

Interface restrictions

Apart from declaration requirements and the type conversions mentioned earlier,

PROLOG terminology

The smallest meaningful program units in PROLOG are numbers and atoms. An atom is a word beginning with a lowercase letter. All words beginning with uppercase letters are treated as variables, which may hold atoms, numbers, or larger constructs.

PROLOG, like LISP, is not a strongly typed language, although the type of a value can generally be determined and used in a PROLOG rule. Variables that have values are referred to as being instantiated.

A PROLOG structure consists of a functor and a parenthesized series of comma-separated arguments, as in *has(Person,Thing)* or *ishuman-(ronald)*. These constructs are best thought of as denoting relationships.

The fundamental data structure of PROLOG is the term, which encompasses atoms, variables, and structures alike. Depending on its position in a PROLOG program, a term is interpreted as either a piece of data, statement of truth, or goal or query to be deduced.

In pure PROLOG, there is no distinction between a piece of data and a statement of truth. However, PROLOG implementations such as Arity Corp.'s permit the programmer to make a distinction to gain data base efficiency and functionalities.

Terms that do not involve variables (such as *ishuman(ronald)*) are called ground. When PROLOG execution finds a nonground goal, it tries to reduce it to a ground instance of that goal by finding a consistent substitution for the variables in the goal's arguments.

For example, the goal *max(X,Y,X)* might succeed with the substitution, $X=10$ and $Y=5$, yielding *max(10,5,10)*. *max(5,10,10)* would be inconsistent in trying to match X simultaneously to both 5 and 10.

A PROLOG program consists of a set of rules for deducing the truth of a given goal from a conjunction of other goals. These rules (clauses) are of the general form:

```
goal1 :- goal2, goal3, ..., goalN
```

Such a rule would be read as "*goal1* is true (successful) if *goal2* can be established, *goal3* can be established, ..., and *goalN* can be established such that all variable substitutions (within and between goals) are consistent."

In this example, *goal1* is said to be the head of the rule and the conjunction of goals, and *goal2, goal3, ...* is said to be the body.

A set of rules with the same head represents alternative ways of establishing the same goal. Such a set is sometimes called a procedure. An example of a procedure consisting of two rules is:

```
max(X,Y,X) :- number(X), number(Y),
  X >= Y
max(X,Y,Y) :- number(X), number(Y),
  Y > X
```

Note that a head is part of the same procedure as another head if they have the same functor and number of arguments. The number of arguments of a term is called the arity of that term.

PROLOG procedures are somewhat analogous to procedures in mainstream languages and are conceptually like *CASE* (or switch) statements (the rules being analogous to branches). The goals in the body of a procedure are invoked sequentially, and a procedure is exited when all goals in one of its rules have succeeded.

However, a fundamental difference in the flow of control arises when one of the goals fails. Whereas in mainstream languages failed statements must be handled explicitly, PROLOG automatically backs up to the previous goal, attempting to satisfy it in a different way. This behavior is known as backtracking and is key to understanding the language.

The cut operator—an exclamation point (!)—is a controversial and important topic in practical applications of the PROLOG language. Its effect is to inhibit backtracking. When placed between two goals in the body of a rule, the cut operator will prevent execution from reattempting the first goal when the second one fails. Instead, the entire procedure will fail.

While overuse of this construct can have insidious results, judicious application of the cut can dramatically increase a program's storage and runtime efficiency.

New

C Applications
In HOURS - Not Months!
with Quic-PRO 5™

Revolutionary new software lets you write applications with a powerful, high level, 4th generation development system that translates 100% to Language C.

Unique applications code generator builds complete programs from fill-in-the-blanks specifications and "painted" screens. Even non-programmers can use it to build complete applications with menus and help screens. Skilled programmers delight in the speed of creation of basic application elements.

Comprehensive, 4th generation, structured, interpretive, transaction processing oriented, database language to polish up the programs created on the applications code generator or create new ones from scratch. Over 10,000 developers, worldwide, have used this powerful, fully debugged, product as Q-PRO 4. You too can develop industrial grade, complex applications in a remarkably short time. Professionals love its over 100 commands, data dictionary, path support, event driven architecture, memory and file arrays, table lookup and other advanced features.

Unbelievably fast, multikey B+ tree, VSAM/ISAM file system makes your applications run blazingly fast. No limits either -- on number of fields, record size, file size or number of files open at one time.

Complete recording with the easi-to-use, multifile query language or the uniquely powerful report generator. Or write your own.

100% translation of executable program code to K & R compatible Language C. After you have your entire application working, tested and debugged, convert it to C. Even if you have never written a C program before! And, use the same file handler. Translation is to Datalite C (standard), Lattice C or MicroSoft C.

Compile and link using the included (standard version only), and fully documented, Datalite C and linker.

Fantastic results! Complete, fast running compiled applications -- quickly, easily!

LAN and Multiuser versions fully support Novell Netware, IBM PC NetWork, Concurrent DOS and MUCDOS file and record locks.

Unbelievable value. Complete system only $199.95 for the single user, small memory model; $595 for the LAN (Novell, IBM PC Network) all memory models. Full 30 day money back guarantee.

Requires PC/MS-DOS, 512 ram, hard disk.

Q-N-E international

136 Granite Hill Court
Langhorne, PA 19047 (215) 968-5966

© 1987. Q-N-E International.
TLX 291765

CIRCLE 21 ON READER SERVICE CARD

two restrictions imposed by Arity's PROLOG-to-C interface necessitated accommodations in the structure of our sample program.

The first was that C could not invoke PROLOG. The main implication of this for the PROLOG side was that all information regarding interaction with the user had to be passed back to PROLOG through preestablished functions before some actions could be taken.

For example, the C functions, not being able to access the PROLOG data base directly, could not directly invoke a PROLOG procedure to return a list of names. If for some reason C needs to retain data in between communication events with PROLOG, local variables cannot be used since the function must exit before communication can take place.

The implementation of the *list* command was designed to illustrate how C could retain data in this way by using a dynamically allocated buffer. This strategy can be further generalized by using an array of such buffers.

The second restriction has to do with the behavior of C functions during backtracking. As noted earlier, when a PROLOG goal fails, previously encountered goals are reinvoked in an attempt to find a different way to satisfy them. If successful, this process of reentering goals often results in a different instantiation of the arguments to that goal.

Retrieving items from a data base is an example of a goal that typically succeeds during backtracking unless all items have already been tried. A goal that invokes a random-number generator should always succeed during backtracking since a new random number can always be generated.

An example of the use of backtracking is the second rule of the *even_or_odd_num_inbetween* procedure. The third goal in that rule invokes the *inbetween* procedure to generate a random number within the specified bounds. The last goal then tests that the resultant number is even or odd, as required by the instantiation of the *EvenOdd* argument.

If this test fails, execution will backtrack and reinvoke the third goal, resulting in the generation of a new random number, which is then tested, and so forth, until the test can be satisfied. (Note that while the use of backtracking here is largely for illustrative reasons, more complex tests could easily be substituted in place of even or odd.)

Arity's v. 4.0, however, does not support reinvocation of C goals. Instead, they simply fail automatically during backtracking. One method of working around this limitation is illustrated in the *found_random* rule used by *inbetween* to generate a random integer. (The procedure *prompt_for_bounds* is a good example also.)

In the definition of *found_random,* the C goal is preceded by a *repeat* goal, which is defined to always succeed. The effect is to reinvoke the C goal immediately after it is passed up during backtracking. This solution works, but it is inappropriate if the C goal can legitimately fail since such failure would cause an infinite loop.

An interesting exercise is to develop a more generic solution. One approch is to replace the *repeat* goal with a *repeat_unless,* which is designed to succeed during backtracking unless a termination symbol has been recorded in the PROLOG data base.

The C function then should be defined such that it invariably succeeds but has an extra parameter to serve as a flag, signalling when failure is intended. An *end_repeat* procedure must test this flag and, when indicated, fail in place of the C function—but not before recording the termination symbol in the data base so that *repeat_unless* can also fail.

The detailed requirements for interfacing Arity/PROLOG and Microsoft C are spelled out in the Arity Corp. booklet *Building Arity/PROLOG Applications.* Among other things, this publication goes into the requirements for compiling and linking.

My one observation about these requirements is that it pays to use a full-featured make utility (such as PS-MAKE, distributed by Unipress) and to save your make files for future reference.

Several additional files are available on the COMPUTER LANGUAGE *Bulletin Board Service and CompuServe forum.*

References

Clocksin, W.F., and C.S. Mellish. *Programming in PROLOG,* 2nd Ed. New York: Springer-Verlag, 1984.

Sterling, Leon, and Ehud Shapiro. *The Art of PROLOG: Advanced Programming Techniques.* Cambridge, Mass.: The MIT Press, 1986.

The Arity/PROLOG Programming Language and *Building Arity/PROLOG Applications,* Arity Corp., Concord, Mass., 1986.

M. Haytham Matthews is an associate of Information Resource Management Associates Inc. He is currently serving as a consultant on business applications of fifth generation languages and tools in New York, N.Y.

```
%              T H E    P R O L O G    M O D U L E              %

%   Arity requires that the certain procedures be declared     %
%   public and/or visible.  In addition, procedures may be     %
%   used from other files, so long as they are declared        %
%   "extrn" and linked in. The "extrn" declarations for        %
%   C functions also have a special format. The number         %
%   following each declaration is the arity of the procedure,  %
%   (the number of arguments it takes).                        %
:- public main/0.
:- visible main/0.

:- extrn c_dispatch_3/3:c('_c_dispatch_3').
:- extrn c_dispatch_2/2:c('_c_dispatch_2').
:- extrn c_init/0:c('_c_init').

%   A top-level procedure called "main" must be included in    %
%   compiled Arity PROLOG programs.  It is called auto-        %
%   matically when the program is started.  The following      %
%   "main" procedure contains an iterative loop that prompts   %
%   for a command and then causes an appropriate action to     %
%   be taken.                                                  %
main :-
        c_init,
        perform(help),
        repeat,
            performed_by_c(queryuser,Command),
            perform(Command),
        Command = = bye.

%   The dispatch procedures used here are imperative in        %
%   nature. The cut operator (!) is appended in the next       %
%   example to explicitly preclude any attempt to reenter      %
%   dispatch goals during backtracking.                        %
perform(A) :-
        dispatch(A),!.

performed_by_c(A,B,C) :-
        c_dispatch_3(A,B,C),!.

performed_by_c(A,B) :-
        c_dispatch_2(A,B),!.

%   dispatch is a procedure that responds to commands          %
%   received from the user.  The last clause serves as a       %
%   default action.                                            %
dispatch(Command) :-
        Command = = random,!,
        prompt_for_bounds(B1,B2),
        prompt_for_even_or_odd(EO),
        even_or_odd_num_inbetween(N,EO,B1,B2),
        performed_by_c(telluser,N).
dispatch(Command) :-
        Command = = name,!,
        new_name_is(First,Middle,Last),
        save_and_sort(First,Middle,Last).
dispatch(Command) :-
        Command = = list,!,
        lastname_starts_with(Prefix),
        list_names_starting_with(Prefix).
dispatch(Command) :-
        Command = = bye,!,
        save,
        performed_by_c(telluser,'Bye!').
dispatch(Command) :-
        Command = = help,!,
        performed_by_c(telluser,
              'COMMANDS:  name, list, random, bye, help.'),
        performed_by_c(telluser,
              'NAME:  add a new name to the database.'),
        performed_by_c(telluser,
              'LIST:  list names in the database.'),
        performed_by_c(telluser,
              'RANDOM: generate an even or odd random number.'),
        performed_by_c(telluser,
              'BYE:   save names to the database and quit.'),
        performed_by_c(telluser,
              'HELP:  display this message.').
```

Listing 1. *(Continued on following page)*

New Release!

SEIDL MAKE UTILITY

Version 2.0

The BEST just got BETTER!

If you're serious about software development, consider the **SEIDL MAKE UTILITY** (*SMK*). *SMK* is not just another copy of the Unix Make. It was specifically designed to deliver features and performance not found in other makes.

✓ ***Structured Language*** to describe dependencies in a clear, concise and portable manner.

✓ ***Rich Command Set*** includes parameterized macros, variables, if-then-else, iteration, wild cards, exception cases, macro libraries, interactive statements, environment access, pattern matching and much more!

✓ ***Intelligent Analysis*** algorithm handles nested include files, library dependencies, and performs consistency tests to detect errors that other makes would blindly ignore.

✓ ***Seidl Version Manager*** compatibility lets you expand your system into the most comprehensive revision/version control system available.

"SMK is a very good Make indeed. Its major distinction is a truly simple Dependency Definition Language, easy to learn and easy to use... you'll probably bless SMK."
— Sextant, July '86

"SMK offers many unique features. [The error handling facility] is extremely useful if a large number of files must be recompiled."
— Computer Language, June '86

DOS Version Only **$99.95** $3.50 p&h Call for other op systems.

Call Today
1-313-662-8086
Visa/MC/COD Accepted
Dealer Inquiries Invited

SEIDL COMPUTER ENGINEERING
3106 Hilltop Dr., Ann Arbor, MI 48103

CIRCLE 22 ON READER SERVICE CARD

DIRECTLY ACCESS dBASE III DATA FILES

db DOS $39.95

CREATE, VIEW AND EDIT dBase III data files from the DOS prompt. Use fast and powerful full screen editing with search, append and direct GOTO capabilities. Output to the screen or printer at your choice. All this from the DOS prompt.

db PASCAL $29.95

TURBO CHARGE YOUR TURBO PASCAL FILE ACCESS by using dBase III data files. Simple dBase III file access allows report output with turbo speed. Includes a create utility which writes record structure code automatically. This allows field access using dBase field names. Sample programs provided and source code is included.

ORDER NOW

Phone orders

1-800-433-6854

In Arizona

(602) 435-2370

Or send check or money order to:
LogicPath P.O. Box 1267
Chandler, Arizona 85224-1267.

We welcome Visa/MC or COD in certified US funds only. Add $2.50 for shipping per order. In Arizona add 6% sales tax. Call or write for other products or additional information (602) 435-2370.

LOGICPATH

P.O. Box 1267 • Chandler, AZ 85244-1267

dBase III is a trademark of Ashton-Tate.
Turbo Pascal is a trademark of Borland Intl.

CIRCLE 23 ON READER SERVICE CARD


```
dispatch(Default) :-
          performed_by_c(telluser,
                    'COMMANDS:  name, list, random, bye, help.').

%   prompt_for_bounds succeeds when the user enters two   %
%   positive numbers to be used as the bounds for         %
%   generating a random number.                           %
prompt_for_bounds(Bound1,Bound2) :-
          repeat,
            performed_by_c(promptforint,
       'Please give me an integer upper or lower bound: ',Bound1),
            performed_by_c(promptforint,
       'Please give me the other bound: ',Bound2),!.

% prompt_for_even_or_odd succeeds on user input of a 1    %
% (odd) or Ø (even)                                      %
prompt_for_even_or_odd(E) :-
          repeat,
                performed_by_c(promptforint,
                'Number should be odd (1) or even (Ø): ',E),
          (E = = Ø; E = = 1),!.       % semicolons are       %
                                      % interpreted as "or"  %

%   even_or_odd_num_inbetween succeeds by testing for or  %
%   generating an integer within the two (given) bounds   %
%   that is even if EvenOdd is Ø or odd if EvenOdd is 1.  %
%   It is interesting to generalize this procedure to     %
%   generate a number evenly divisible by an arbitrary    %
%   integer (not just 2)                                  %
even_or_odd_num_inbetween(SameNum,EvenOdd,SameNum,SameNum) :-
          (EvenOdd = = Ø; EvenOdd = = 1),
          EvenOdd is abs(SameNum) mod 2.
even_or_odd_num_inbetween(N,EvenOdd,Bound1,Bound2) :-
          (EvenOdd = = Ø; EvenOdd = = 1),
          ordered(Bound1,Bound2,LwrBound,UprBound),
          UprBound - LwrBound > Ø,
          inbetween(N,Bound1,Bound2),
          EvenOdd is abs(N) mod 2.

%   ordered unifies its second two arguments with its     %
%   first two. On success the second two arguments are    %
%   instantiated and in ascending order.                  %
ordered(N1,N2,N1,N2) :- N2 >= N1.
ordered(N1,N2,N2,N1) :- N2 < N1.

%   inbetween succeeds if its 1st argument is in the open %
%   interval defined by its 2nd two arguments or, failing %
%   that, by generating a random number and applying a    %
%   transformation to force it into the interval.         %
inbetween(N,Bound1,Bound2) :-
          nonvar(N),
          in_interval(N,Bound1,Bound2).
inbetween(N,Bound1,Bound2) :-
          var(N),
          found_random(Num),
          placed_between(Num,Bound1,Bound2,N).

%   placed_between applies a tranformation to its first   %
%   argument to generate a new number within the bounds   %
%   specified by its 2nd and 3rd arguments.               %
placed_between(N,Bound1,Bound2,Nplaced) :-
          ordered(Bound1,Bound2,LwrBound,UprBound),
          IntrvlSize is UprBound - LwrBound + 1,
          Nmod is N mod IntrvlSize,
          Nplaced is abs(Nmod) + LwrBound.

%   in_interval succeeds if its 1st argument is within    %
%   the open interval defined by its 2nd two arguments.   %
in_interval(N,Bound1,Bound2) :-
          ordered(Bound1,Bound2,LwrBound,UprBound),
          N >= LwrBound,
          N =< UprBound.
```

Listing 1. *(Continued on following page)*

```
%   found_random generates a random number.  So long as   %
%   its first argument is an uninstantiated variable,      %
%   it always succeeds.  repeat is used to simulate        %
%   reentering the C goal during backtracking.             %
found_random(N) :-
        var(N),
        repeat,
        performed_by_c(random,N).

new_name_is(First,Middle,Last) :-
        performed_by_c(promptfortext,'First name: ',First),
        performed_by_c(promptfortext,'Middle Name: ',Middle),
        performed_by_c(promptfortext,'Last name: ',Last).

%   save name in a b-tree sorted by lastname, firstname,   %
%   middlename                                             %
save_and_sort(FirstName,MiddleName,LastName) :-
      concat([LastName,FirstName,MiddleName],Sortkey),
      recordb(nametree,Sortkey,fullname(FirstName,MiddleName,LastName)).

lastname_starts_with(Letters) :-
     performed_by_c(promptfortext,
       'List names where last name starts with (* = anything): ',Letters).

% invoke any_more_names_to_output to display names in the  %
% data base. Succeeds by negating any_more_names_to_output %
% (the latter is guaranteed to fail).                      %
list_names_starting_with(Letters) :-
        Letters = = '*',!,
        not any_more_names_to_output.
list_names_starting_with(Letters) :-
        Letters \= '*',!,
        not any_more_names_to_output(Letters).

%   The rules below backtrack over the b-tree outputting   %
%   names to the screen.  They ultimately fail when no     %
%   more names remain to be displayed. The rule that       %
%   takes a Prefix argument also tests names, displaying   %
%   only those where the last name begins with the         %
%   letters in Prefix.                                     %
any_more_names_to_output :-
    retrieveb(nametree,SortKey,fullname(FirstName,MiddleName,LastName)),
    performed_by_c(holdthis,FirstName),
    performed_by_c(displayname,MiddleName,LastName),
    fail.

any_more_names_to_output(Prefix) :-
    retrieveb(nametree,SortKey,fullname(FirstName,MiddleName,LastName)),
       string_search(Prefix,LastName,Ø),
       performed_by_c(holdthis,FirstName),
       performed_by_c(displayname,MiddleName,LastName),
    fail.
```

Listing 1. *(Continued from preceding page)*

```
/*                    T H E    C    M O D U L E              */

/* These six header files are provided by the Microsoft C  */
/* compiler and contain declarations for C library         */
/* functions used in the program.                          */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <malloc.h>
#include <limits.h>

/* The header file apctype.h provided by Arity contains
   declarations for "reftype" and for the Arity library
   functions that handle conversions between PROLOG and
   C data types.  SUCCESS and FAILURE are also defined
   as integer constants in apctype.h.  C functions invoked
   by PROLOG are only allowed to return one of these values.
   Other values must be passed back through the arguments
   in standard PROLOG fashion.                              */
```

Listing 2. *(Continued on following page)*

NEW FROM
C WARE

TA-DAH!

DESMET C
V 3.0

THE FASTEST
C COMPILER JUST GOT
FASTER!

FASTER DEVELOPMENT

Invoke DeSmet C from our SEE™ editor. The first error returns you immediately to SEE at the error line with the error message displayed.

FASTER COMPILATION

When you don't use inline assembly code or don't want to see the ASM88 output, the V3.0 compiler produces object code directly – making DeSmet C up to *twice* as fast as before.

PLUS

EXPANDED STANDARD LIBRARY

New environment, networking, path, file, time and string functions now included.

DESMET GAZETTE

While our supply lasts, we are shipping all new customers an introductory copy of this excellent code intensive quarterly published by Pacific Data Works.

C88 C Compiler Still only $109
D88 Debugger 50
Large Case Option 50
MAKE utility (an excellent value)50

Product literature sent on request.

Call us now and we'll ship today.

C WARE
CORPORATION
(408) 720-9696 TELEX: 358185
P.O. Box C
SUNNYVALE, CA 94087 USA
VISA, MC & AMEX
Street Address: 505 W. Olive Suite 767
Tech BBS (408) 720-8197
300-1200-2400 BAUD

MetaWINDOW

Power Graphics for your PC!

PC TECH JOURNAL

"Product of the Month"
"... a technological tour de force for fast PC graphics."

NO ROYALTIES!

MetaWINDOW is an advanced, high performance graphics development toolkit which bridges the gap between low-level graphic primitive libraries and pre-packaged window managers.

Unparalleled Performance!

MetaWINDOW provides an expanded set of graphic drawing functions, plus the added functionality and performance required for designing multi-window desktop applications.

- *auto-cursor tracking*
- *pull-down menus*
- *pop-up windows*
- *comprehensive graphic functions*
- *multiple fonts*

Enhanced Features!

- *Display multiple bitmap or "filled-outline" fonts.*
- *Face fonts for bold, italic, underline or strike-out stylings.*
- *Full "RasterOp" transfer functions for writing, erasing, rubberbanding or dragging: lines, text, icons, bit images and complex objects.*
- *Create pop-up menus, windows and icons.*
- *Supports IBM's new PS/2 VGA and MCGA graphics.*

MetaWINDOW comes complete with langauge bindings for 16 popular C, Pascal and Fortran compilers, plus dynamic runtime support for over 40 graphics adaptors and input devices.

MetaWINDOW
Advanced Graphics Toolkit
*4 disks, 3 260 page manuals - $195**

NEW! - TurboWINDOW/C
*All the features of MetaWINDOW for Borland Turbo C! - $ 95**
** Plus $5.00 shipping and handling*

TO ORDER CALL 1-800-332-1550
For information or in CA call 408-438-1550

METAGRAPHICS
SOFTWARE CORPORATION
269 Mount Hermon Road
Scotts Valley, CA 95066

CIRCLE 24 ON READER SERVICE CARD


```
#include "apctype.h"

#define YES 1
#define NO Ø

/* STREQ compares two strings for equality */
#define STREQ(a,b) (stramp(a,b)= =Ø)

static char Genbuf[1Ø24];    /* general-purpose buffer for */
                             /* transient storage          */

static struct BUFTYPE {       /* dynamically sized buffer       */
                char *buf;    /* whose storage space allocation */
                int bufinuse; /* and deallocation are handled   */
                } Hold;       /* by stash() and release()       */

int c_init()
{
     Hold.bufinuse = NO;
     return(SUCCESS);
}

/* These are the two dispatch functions that administer
commands received from PROLOG.  As a convention, the
number of arguments they take (their arity in PROLOG
terminology) is appended to the function names.          */
int c_dispatch_2(cmd,arg)
reftype cmd,arg;
{
     int length;
     char *command = Genbuf;

/* "findtype_c" is an Arity function that returns
an integer indicating the data type of its PROLOG
argument via defined constants (ATOM, etc.)     */
     if (findtype_c(cmd) != ATOM)
          return(FAILURE);

     gettxt_c(cmd,command,&length); /* "gettxt_c" converts a */
                                    /* PROLOG atom or string */
                                    /* to a C array of char. */
     if (STREQ(command,"queryuser"))
          return(query_user(arg));
     if (STREQ(command,"random"))
          return(random(arg));
     if (STREQ(command,"telluser"))
          return(telluser(arg));
     if (STREQ(command,"holdthis"))
          return(holdthis(arg));
     return(FAILURE);
}
int c_dispatch_3(cmd,arg,arg1)
reftype cmd,arg,arg1;
{
     int length;
     char *command = Genbuf;

     if (findtype_c(cmd) != ATOM)
          return(FAILURE);
     gettxt_c(cmd,command,&length);
     if (STREQ(command,"promptforint"))
          return(promptfor(INTEGER,arg,arg1));
     if (STREQ(command,"promptfortext"))
          return(promptfor(ATOM,arg,arg1));
     if (STREQ(command,"displayname"))
          return(displayname(arg,arg1));
     return(FAILURE);
}

/* Generate a random number and return it in PROLOG format.  */
int random(number)
reftype number;
{
     if (findtype_c(number) != UNDEF) /* UNDEF is a defined */
          return(FAILURE);       /* constant meaning an uninstantiated */
                                 /* (value-less) variable.  */
     putint_c(rand(),number); /* "putint_c" converts a C */
     return(SUCCESS);         /* integer to PROLOG */
}
```

Listing 2. *(Continued on following page)*

```
/* "telluser" displays its argument to the terminal screen. */
int telluser(arg)
reftype arg;
{
     int number_of_chars, num;

     switch (findtype_c(arg)) {
          case ATOM:
          case STRING:  gettxt_c(arg,Genbuf,&number_of_chars);
                        printf("\n%s\n",Genbuf);
                        return(SUCCESS);
          case INTEGER: getint_c(arg,&num);
                        printf("\n%d\n",num);
                        return(SUCCESS);
          default:      return(FAILURE);
     }
}

/* "promptfor" displays its second argument as a prompt
to the user.  It then accepts input from the user and
succeeds if the input is valid. */
int promptfor(typeneeded,prompt,returnval)
int typeneeded;
reftype prompt,returnval;
{
     int prompttype, length;
          long usrval;
     char *endptr;

     prompttype=findtype_c(prompt);
     if (findtype_c(returnval) != UNDEF ||
        (prompttype != ATOM && prompttype != STRING))
          return(FAILURE);
     gettxt_c(prompt,Genbuf,&length);
     printf("\n%s",Genbuf);
     fflush(stdin);
     if (typeneeded = = INTEGER) {
          scanf("%6s",Genbuf);
          usrval = strtol(Genbuf,&endptr,10);
          if (*endptr != NULL || labs(usrval) > INT_MAX)
               return(FAILURE);     /* fail on invalid input */
          putint_c((int)usrval,returnval);
          return(SUCCESS);
     }
     if (typeneeded = = ATOM) {
          scanf("%20s",Genbuf);
          putatm_c(strlwr(Genbuf),strlen(Genbuf),returnval);
          return(SUCCESS);
     }
     return(FAILURE);
}

/* Store copy of "text" in the dynamically sized global buffer */
int stash(text)
char *text;
{
      char *calloc_failed = NULL;

      if (Hold.bufinuse = = YES)
           return(FAILURE);
      Hold.buf = calloc(strlen(text)+1,sizeof(char));
      if (Hold.buf = = calloc_failed)
          return(FAILURE);
      strcpy(Hold.buf,text);
      Hold.bufinuse = YES;
              return(SUCCESS);
}

/* Free storage in the dynamically sized global buffer */
int release()
{
      if (Hold.bufinuse = = YES) {
           free(Hold.buf);
           Hold.bufinuse = NO;
      }
      return(SUCCESS);
}
```

Listing 2. *(Continued on following page)*

TRUE
MULTITASKING
With
MultiDos Plus
"multitasking for the IBM-PC."

Ideal for developing applications in process control, data acquisition, communications, and other areas. Check these features which make **MultiDos Plus** an unbeatable value.

- Run up to 32 programs concurrently.
- Your software continues to run under DOS. No need to learn a new operating system.
- Use the compilers you already have. Supports software written in most languages.
- Operator commands to load/run programs, change priority, check program status, abort/suspend/resume programs.
- Programmatic interface via INT 15H for the following.
  - * Intertask message communication. Send/receive/check message present on 64 message queues.
  - * Task control by means of semaphores. Get/release/check semaphores.
  - * Change priority-256 priority levels.
  - * Suspend task for specified interval.
  - * Spawn and terminate external and internal tasks.
  - * Disable/enable multitasking.
  - * and more!
- Independent foreground/background displays.
- Access to DOS while applications are running.

Hardware/Software Requirements

IBM PC/XT/AT or true clone. Enough memory to hold **MultiDos Plus** (48 KB) and all your application programs. Also may need 4 or 16 KB memory for "hidden screens" for each active task. MS-DOS (or PC-DOS) 2.0 or later operating system.

only: **$24.95** *OR*
$99.95
with source code

Outside USA add $5.00 shipping and handling.
Visa and Mastercard orders only call toll-free: 1-800-872-4566, ext. 350., or send check or money order to:

NANOSOFT
13 Westfield Rd, Natick, MA 01760
MA orders add 5% sales tax.

CIRCLE 25 ON READER SERVICE CARD

COMBINE THE
RAW POWER OF FORTH
WITH THE CONVENIENCE
OF CONVENTIONAL LANGUAGES

HS/FORTH

Now you can compile even large programs in the blink of an eye. If you don't need to compile the huge fast programs we handle so well, use the metacompiler to spin off compact ones — as small as a few hundred bytes for simple threaded utilities — as small as 2 kbytes for a full Forth execution core. HS/FORTH is the best base from which to spin off either direct or indirect threaded systems, small or large, or anything else you might invent. This is absolutely the most flexible Forth system available, at any price, with no expensive extras you'll need to buy later. Distribute metacompiled tools, or turnkeyed applications royalty free.

Although HS/FORTH is unmatched for language experimentation and development, remember that we wrote it to be a top notch application development system. Your application will have all of DOS at its disposal, commands, other programs, functions, everything! I/O is easier than in Pascal or Basic, but much more powerful — whether you need parsing, formatting, or random access. Send display output through DOS, BIOS, or direct to video memory. Windows organize both text and graphics display. Math facilities include both software and hardware floating point plus an 18 digit integer (finance) extension and fast arrays for all data types. The hardware floating point covers the full range of trig, hyper and transcendental math including complex. Forth gives you total control of your computer, but only HS/FORTH gives you implemented functionality so you aren't left hanging with "great possibilities" (and lots of work!)

We can't possibly cover everything in this ad, so please call or write and ask for our brochure. We'll also be happy to answer any questions.

HS/FORTH complete system:	$395.
Forth: Text & Reference (500pg) Kelly&Spies, Prentice Hall	$ 22.
HS/FORTH: Tutorial & Reference Kelly&Callahan (500pg)	$ 59.
HS/FORTH Glossary	$ 10.
DEMO DISK	$ 10.

VISA Visa Mastercard MasterCard

HARVARD SOFTWORKS
PO BOX 69
SPRINGBORO, OH 45066
(513) 748-0390

CIRCLE 26 ON READER SERVICE CARD

```
/* Save a copy of "text" using stash() to allocate storage  */
int holdthis(text)
reftype text;
{
     int length, texttype;

     texttype = findtype_c(text);
     if (texttype != ATOM && texttype != STRING)
          return(FAILURE);
     gettxt_c(text,Genbuf,&length);
     return(stash(Genbuf));
}

/* "displayname" displays a full name to the screen. It is
    designed to illustrate use of dynamically sized global
    buffers to store data in between returns to PROLOG.
    Here, the PROLOG procedures "any_more_names_to_output"
    invoke the "holdthis" function to save the first name
    in a global buffer.  Then "displayname" is invoked
    with the middle and last names, so that all three are
    available for simultaneous output.  Note that this
    approach is used here for didactic reasons only since
    a more straightforward solution here would have been
    to define a new dispatch function, "c_dispatch_4,"
    that accepted four arguments while maintaining the
    PROLOG-to-C communication convention.*/
int displayname(middlename,lastname)
reftype middlename,lastname;
{
     int length;
     char *middle=Genbuf, last[80];

          if (Hold.bufinuse = = NO)   /* Fail if the first */
          return(FAILURE);          /* name is not in Hold.buf */
     gettxt_c(middlename,middle,&length);
     gettxt_c(lastname,last,&length);
     middle[0] = toupper(middle[0]);
     last[0] = toupper(last[0]);
     Hold.buf[0] = toupper(Hold.buf[0]);
     printf("\n     %-20s     %-20s     %-20s",Hold.buf,middle,last);
     release();
     return(SUCCESS);
}

/* Issue the top-level prompt and return the user response
   as a PROLOG atom. */
int query_user(command)
reftype command;
{
     if (findtype_c(command) != UNDEF)
          return(FAILURE);
     printf("\nCMD>");
     fflush(stdin);
     scanf("%6s",Genbuf);
     putatm_c(strlwr(Genbuf),strlen(Genbuf),command);
          /* putatm_c converts a C array of char */
          /* to a PROLOG atom. */
     return(SUCCESS);
}
```

Listing 2. *(Continued from preceding page)*

SAS Institute Inc. Announces

Lattice C Compilers for Your IBM Mainframe

Two years ago...

SAS Institute launched an effort to develop a subset of the SAS® Software System for the IBM Personal Computer. After careful study, we agreed that C was the programming language of choice. And that the Lattice® C compiler offered the quality, speed, and efficiency we needed.

One year ago...

Development had progressed so well that we expanded our efforts to include the entire SAS System on a PC, written in C. And to insure that the language, syntax, and commands would be identical across all operating systems, we decided that all future versions of the SAS System—regardless of hardware—would be derived from the same source code written in C. That meant that we needed a C compiler for IBM 370 mainframes. And it had to be good, since all our software products would depend on it.

So we approached Lattice, Inc. and asked if we could implement a version of the Lattice C compiler for IBM mainframes. With Lattice, Inc.'s agreement, development began and progressed rapidly.

Today...

Our efforts are complete—we have a first-rate IBM 370 C compiler. And we are pleased to offer this development tool to you. Now you can write in a single language that is source code compatible with your IBM mainframe and your IBM PC. We have faithfully implemented not only the language, but also the supporting library and environment.

Features of the Lattice C compiler for the 370 include:

- **Generation of reentrant object code.** Reentrancy allows many users to share the same code. Reentrancy is not an easy feature to achieve on the 370, especially if you use non-constant external variables, but we did it.
- **Optimization of the generated code.** We know the 370 instruction set and the various 370 operating environments. We have over 100 staff years of assembler language systems experience on our development team.
- **Generated code executable in both 24-bit and 31-bit addressing modes.** You can run compiled programs above the 16 megabyte line in MVS/XA.
- **Generated code identical for OS and CMS operating systems.** You can move modules between MVS and CMS without even recompiling.
- **Complete libraries.** We have implemented all the library routines described by Kernighan and Ritchie (the informal C standard), and all the library routines supported by Lattice (except operating system dependent routines), plus extensions for dealing with 370 operating environments directly. Especially significant is our byte-addressable Unix®-style I/O access method.
- **Built-in functions.** Many of the traditional string handling functions are available as built-in functions, generating in-line machine code rather than function calls. Your call to move a string can result in just one MVC instruction rather than a function call and a loop.

In addition to mainframe software development, you can also use our new cross-compiler to develop PC software on your IBM mainframe. With our cross-compiler, you can compile Lattice C programs on your mainframe and generate object code ready to download to your PC.

With the cross-compiler, we also offer PLINK86™ and PLIB86™ by Phoenix Software Associates Ltd. The Phoenix link-editor and library management facility can bind several compiled programs on the mainframe and download immediately executable modules to your PC.

Tomorrow...

We believe that the C language offers the SAS System the path to true portability and maintainability. And we believe that other companies will make similar strategic decisions about C. Already, C is taught in most college computer science curriculums, and is replacing older languages in many. And almost every computer introduced to the market now has a C compiler.

C, the language of choice...

C supports structured programming with superior control features for conditionals, iteration, and case selection. C is good for data structures, with its elegant implementation of structures and pointers. C is conducive to portable coding. It is simple to adjust for the size differences of data elements on different machines.

Continuous support...

At SAS Institute, we support all our products. You license them annually; we support them continuously. You get updates at no additional charge. We have a continuing commitment to make our compiler better and better. We have the ultimate incentive—all our software products depend on it.

For more information...

Complete and mail the coupon today. Because we've got the development tool for your tomorrow.

SAS®

SAS Institute Inc.
SAS Circle, Box 8000
Cary, NC 27511-8000
Telephone (919) 467-8000 x 7000

I want to learn more about:

- ☐ the C compiler for MVS software developers
- ☐ the C compiler for CMS software developers
- ☐ the cross-compiler with PLINK86 and PLIB86

today...so I'll be ready for tomorrow.

Please complete or attach your business card.

Name _____

Title _____

Company _____

Address _____

City _____ State _____ ZIP _____

Telephone _____

Mail to: SAS Institute Inc., Attn: CC, SAS Circle, Box 8000, Cary, NC, USA. 27511-8000. Telephone (919) 467-8000, x 7000

CL 7/87

SAS is the registered trademark of SAS Institute Inc., Cary, NC, USA. Lattice is the registered trademark of Lattice, Inc. PLINK86 and PLIB86 are trademarks of Phoenix Software Associates Ltd. UNIX is the trademark of AT&T. Copyright © 1985 by SAS Institute Inc. Printed in the USA.

DISCOVER PARADISE

Programmer's Paradise Gives You Superb Selection, Personal Service and Unbeatable Prices!

Welcome to Paradise. The MS/PC-DOS and XENIX software source that caters to your programming needs.

Discover the Many Advantages of Paradise ...

- Lowest price guaranteed
- Latest versions
- Huge inventory, immediate shipment
- Knowledgeable sales staff
- Special orders
- 30-day money-back guarantee*

NEW CLUES THIS MONTH!

Discover Paradise Contest— Search for the Hidden Clues to our Island's Identity and Win!

	LIST	OURS
C++		
ADVANTAGE C++	$ 495	479
PFORCE++	395	215
C COMPILERS		
AZTEC C-86 COMMERCIAL	499	CALL
AZTEC C-86 DEVELOPER	299	CALL
C86 PLUS	497	379
DATALIGHT C DEVELOPER'S KIT	99	79
DATALIGHT—OPTIMUM C	139	109
LATTICE C	500	265
W/SOURCE	900	495
LET'S C	75	59
W/CSD DEBUGGER	150	99
MARK WILLIAMS C	495	289
MICROSOFT C	450	269
MICROTEK 68000/08/10/20 C CROSS	CALL	CALL
TURBO C	100	65
UNIWARE 68020 C CROSS	595	CALL
WIZARD C	450	299
C INTERPRETERS		
C-TERP	300	229
INSTANT C	500	379
INTERACTIVE C	249	219
INTRODUCING C	125	105
Q C	60	55
RUN/C	120	79
RUN/C PROFESSIONAL	250	155
ASSEMBLERS, LINKERS		
ADVANTAGE DISASSEMBLER	295	265
ADVANTAGE LINK	395	359
ASMLIB	149	129
DIS 'N DATA	165	CALL
MACRO ASSEMBLER	150	93
PASM 86	195	109
PLINK 86 PLUS	495	279
QUELO 68000 X-ASM	595	509

FREE SCIENTIFIC & ENGINEERING SOLUTIONS CATALOG

Describes 100 of the best software packages for solving complex equations, number crunching, analyzing data, 3-D CAD/CAM design, technical word processing, circuit design, sophisticated graphics and more. Call for your FREE catalog today!

	LIST	OURS
GRAPHICS		
ADVANTAGE GRAPHICS	250	225
ESSENTIAL GRAPHICS	250	189
GRAPHIC	350	285
GSS GRAPHICS DEVELOPMENT TOOLKIT	495	375
GSS KERNEL SYSTEM	495	375
GSS METAFILE INTERPRETER	395	235
GSS PLOTTING SYSTEM	495	375
HALO—ONE LANGUAGE	300	205
HALO—FIVE MICROSOFT LANGUAGES	595	389
METAWINDOW	195	CALL
METAWINDOW PLUS	275	CALL
METAFONTS	95	CALL
METAFONTS PLUS	275	CALL

	LIST	OURS
C UTILITY LIBRARIES		
ASYNCH MANAGER	175	119
BASIC_C	175	129
BLACKSTAR FUNCTIONS	99	89
C ESSENTIALS	100	79
C-DISPLAY LIBRARIAN	99	79
C FOOD SMORGASBORD	150	95
W/SOURCE	300	179
C TOOLS PLUS	175	119
DISPLAY EXPRESS	99	79
ESSENTIAL C UTILITY	185	119
ESSENTIAL COMMUNICATIONS	185	125
COMMUNICATIONS PLUS	250	189
GREENLEAF COMM	185	125
GREENLEAF FUNCTIONS	185	125
INTERWORK	129	109
LINK & LOCATE	350	315
LONGITUDE	180W	CALL
MULTI-C	149	135
MULTI-COMM	149	135
PFORCE	295	209
SYNERGY DEVELOPMENT TOOLKIT	395	249
THE HAMMER	195	135
TIMESLICER	295	265
W/SOURCE	1000	CALL
TOPVIEW TOOLBASKET	250	179
SCREEN DISPLAY, WINDOWS FOR C		
C WORTHY	295	269
CURSES	125	89
W/SOURCE	250	175
FLASH-UP WINDOWS	90	79
GREENLEAF DATA WINDOWS	225	155
W/SOURCE	395	289
MICROSOFT WINDOWS	99	65
WINDOWS DEVELOPMENT KIT	500	309
ON-LINE HELP	149	105
PANEL	295	215
PANEL PLUS	495	395
POLYDESK III	99	79
QUICKSCREEN	295	265
SCREENPLAY (LATTICE)	150	135
SOFTSCREEN HELP	195	175
VIEW MANAGER	275	179
VITAMIN C	225	179
VC SCREEN	99	79
WINDOWS FOR C	250	199
WINDOWS FOR DATA	395	315
Z VIEW	245	175
FILE MANAGEMENT		
ASMTREE	395	339
BTRIEVE	245	185
XTRIEVE	245	185
REPORT OPTION	145	99
BTRIEVE/N	595	455
XTRIEVE/N	595	455
REPORT OPTION/N	345	269
C-ISAM	225	195
C-TREE	395	315
R-TREE	295	239
C-TREE/R-TREE BUNDLE	650	519
CQL	395	329
DBC III/II	250	169
W/SOURCE	500	359
DBC III PLUS	750	595
W/SOURCE	1500	1185

	LIST	OURS
DB_VISTA	195	139
W/SOURCE	495	399
DB_QUERY	195	155
W/SOURCE	495	399
FABS	150	129
FABS PLUS	195	169
INFORMIX ESQL/C	595	545
INFORMIX 4GL	995	789
INFORMIX SQL	795	639
PHACT	295	265
DEBUGGERS		
ADVANCED TRACE 86	175	119
BREAKOUT	125	89
C-SPRITE	175	129
"CANNIBAL ISLES"	CALL	CALL
CODESMITH 86	145	99
CSD DEBUGGER	75	59
PERISCOPE I	345	295
PERISCOPE II	175	145
PERISCOPE II-X	145	109
PERISCOPE III	995	825
PFIX 86 PLUS	395	229
XVIEW 86	60	49
EDITORS		
BRIEF	195	CALL
DBRIEF	95	CALL
BRIEF/DBRIEF BUNDLE	275	CALL
C SCREEN	75	CALL
CVUE	75	59
W/SOURCE	250	199
EDIX	195	155
EMACS	295	265
W/SOURCE	995	895
EPSILON	195	149
FIRSTIME (C)	295	229
KEDIT	125	99
LSE	125	99
PC/VI	149	109
PMATE	195	115
SPF/PC	195	145
VEDIT	150	99
VEDIT PLUS	185	129
XTC	99	79
SORT UTILITIES		
AUTOSORT	150	129
MICROSOFT SORT	195	129
OPT-TECH SORT	149	109
PROFILERS		
CODE SIFTER	119	89
PFINISH	395	229
THE PROFILER	125	89
LINT		
ISLAND ARC	CALL	CALL
PC LINT	139	99
PRE-C	295	155
MAKE UTILITIES		
ADVANTAGE MAKE	125	99
KORA SEA	CALL	CALL
LMK	195	139
PMAKER	125	79
POLYMAKE	149	119
SAPIENS MAKE	79	75

80386 DEVELOPMENT TOOLS

	LIST	OURS
386 ASM/LINK	495	389
HIGH C	895	759
PROFESSIONAL PASCAL	895	759

	LIST	OURS
TRANSLATORS/BRIDGES		
BASTOC	495	399
BAS_C	85	79
BAS_PAS	85	79
DB2C	299	269
F2C	795	CALL
MB 86	895	CALL
PTRAL	179	CALL
R BRIDGE	395	339
TP2C	249	CALL
ARTIFICIAL INTELLIGENCE		
ARITY COMBO PACK	1095	979
ARITY PROLOG INTERPRETER	295	229
ARITY STANDARD PROLOG	95	79
CHALCEDONY PROLOG	100	89
GOLDEN COMMON LISP	495	CALL
GCL 286 DEVELOPER	1190	CALL
INSIGHT 2+	485	379
IQCLISP	300	CALL
MICROSOFT LISP	250	159
PROLOG 86 PLUS	250	CALL
SMALLTALK V	99	85
TRANS LISP PLUS	195	CALL
TURBO PROLOG	100	65
TURBO PROLOG TOOLBOX	100	65
BASIC		
BETTERBASIC	199	129
BETTERBASIC ADD ONS	CALL	CALL
BETTERTOOLS	95	89
87 QB PAK	69	59
87 SOFTWARE PAK	180	149
EXIM TOOLKIT	50	45
FINALLY	99	89
FINALLY MODULES	99	89
GDPEDIT	99	89
INSIDE TRACK	65	55
MACH 2	75	59
MICROSOFT QUICKBASIC	99	65
PEEKS 'N POKES	45	39
PROFESSIONAL BASIC	99	75
8087 MATH SUPPORT	50	45
PANEL-BASIC	145	115
QUICK PAK	69	59
STAY RES	95	75
TRUE BASIC	150	105
ADD ONS	CALL	CALL
TURBO BASIC	100	65
COBOL COMPILERS/UTILITIES		
COBOL SPII	395	329
FPLIB	149	129
MICROSOFT COBOL	700	439
MICROSOFT SORT	195	129
MICRO/SPF	175	155
OPT-TECH SORT	149	115
REALIA CICS	995	785
REALIA COBOL	995	785
REALIA MENU	150	119

We'll Match Any Nationally Advertised Price.

	LIST	OURS
RM/COBOL	950	649
RM/COBOL 8X	1250	909
SCREENIO	400	CALL
SCREENPLAY	175	129
VISUAL COBOL (MBP)	795	695
FORTRAN COMPILERS/UTILITIES		
ACS TIMES SERIES	495	389
DIAGRAM'ER	129	115
DOCUMENT'ER	129	115
87 SFL	250	225
FORLIB	95	85
FORLIB-PLUS	70	45
FORTRAN SCIENTIFIC SUBROUTINES	295	249
FOR-WINDS	90	69
GRAFLIB	175	159
GRAFMATICS OR PLOTMATICS	135	119
GRAFMATICS AND PLOTMATICS	240	219
LAHEY FORTRAN	477	CALL
MICROSOFT FORTRAN	450	269
NO LIMIT	129	115
PLOTHI	175	159
PLOTHP	175	159
RM/FORTRAN	595	399
STRINGS AND THINGS	70	49
MODULA-2		
FARBWARE MODULA-2	90	79
LOGITECH MODULA-2		
APPRENTICE PACKAGE	99	79
WIZARD'S PACKAGE	199	159
MAGIC TOOLKIT	99	79
ROM PACKAGE & RT DEBUGGER	299	239
WINDOW PACKAGE	49	39
REPERTOIRE	89	75
PASCAL COMPILERS		
MARSHAL PASCAL	189	169
MICROSOFT PASCAL	300	185
PASCAL-2	350	329
TURBO PASCAL	100	65
TOOLS FOR TURBO PASCAL		
ALICE	95	69
FIRSTIME	75	59
FLASH-UP WINDOWS	90	79
SCREENPLAY	100	89
SCREEN SCULPTOR	125	95
T-DEBUG PLUS	60	49
TURBO•ASM	100	85
TURBO ASYNCH PLUS	100	79
TURBO DATABASE TOOLBOX	70	45
TURBO EDITOR TOOLBOX	70	45
TURBO EXTENDER	85	65
TURBO GRAPHIX TOOLBOX	70	45
TURBO HALO	129	99
TURBO NUMERICAL METHODS	100	65
TURBO PROFESSIONAL	70	49
TURBO POWER TOOLS PLUS	100	79
TURBO REF	50	45
TURBO SMITH	69	45
TURBO TUTOR	40	29
TURBO WINDOWS	80	65
OTHER TURBO TOOLS	CALL	CALL
OTHER LANGUAGES		
IF YOU DON'T SEE IT — CALL!!!!!		
APL*PLUS	595	429
JANUS/ADA C PACK	95	89
JANUS/ADA D PACK	900	769
JANUS/ADA ED PACK	395	CALL
LATTICE RPG II COMPILER	750	629
SCREEN DESIGN AID	350	309
SOURCE ENTRY UTILITY	250	199
SORT/MERGE UTILITY	250	199
RPG COMBO	1100	879
PC FORTH	150	109
PC FORTH PLUS	250	199
PERSONAL REXX	125	99
POCKET APL	95	75

	LIST	OURS
BLAISE		
ASYNCH MANAGER	175	119
C TOOLS PLUS	175	119
EXEC	95	69
PASCAL TOOLS	125	99
PASCAL TOOLS 2	100	79
PASCAL TOOLS & PASCAL TOOLS 2	175	119
RUNOFF	50	45
TURBO ASYNCH PLUS	100	79
TURBO POWER TOOLS PLUS	100	79
VIEW MANAGER	275	179
DBASE COMPILERS, RELATED PRODUCTS		
CLIPPER	695	459
DBC III PLUS	750	595
DBRIEF	95	CALL
FOXBASE	395	CALL
FOXBASE PLUS	395	CALL
GENIFER	395	309
QUICK SILVER	599	499
TOM RETTIG'S LIBRARY	100	79
DIGITAL RESEARCH		
CB 86	600	569
PASCAL MT+	400	379
PL1—86	750	709
DR ASSEMBLER + TOOLS	200	189
ACCESS MANAGER	400	379
DISPLAY MANAGER	500	475
FORTRAN 77	350	329

NEW PRODUCTS

ADVANTAGE Disassembler— Provides immediate feedback as you work, storing results in tables on disk. Final output is ready for MS assembler. Handles .COM and .EXE files; 8086/186/286 code and 8087/287 coprocessors.
List $295 **Ours $265**

ADVANTAGE Graphics— Lightning-fast high-performance C language graphics toolkit. Supports over 30 graphics adaptors. Over 200 functions, including an extensive set for building multi-window applications. Object-oriented concepts provide workstation-like capabilities on your PC/XT/AT — yet no royalties!
List $250 **Ours $225**

Periscope III— Real-time hardware breakpoint debugger. Find errors in real-time systems, stop intermittent failures, interface with undocumented systems and eliminate bottlenecks in your code. Includes board, break-out switch, software and manual.
List $995 **Ours $825**

Turbo C— Long-awaited C compiler from Borland. Complete interactive development environment. ANSI C compatible; includes editor, MAKE, pull-down menus and windows.
List $100 **Ours $65**

	LIST	OURS
LATTICE		
LATTICE C COMPILER	500	265
W/SOURCE	900	495
C CROSS REFERENCE GENERATOR	50	39
C FOOD SMORGASBORD	150	95
W/SOURCE	300	179
C-SPRITE	175	129
CURSES	125	89
W/SOURCE	250	175
DBC III/II	250	169
W/SOURCE	500	359
DBC III PLUS	750	595
W/SOURCE	1500	1185
LMK	195	139
SECRETDISK	120	89
SIDETALK	120	89
SSP/PC	350	269
TEXT MANAGEMENT UTILITIES	120	89
TOPVIEW TOOLBASKET	250	179

	LIST	OURS
LIFEBOAT		
ADVANTAGE C++	495	479
ADVANTAGE DISASSEMBLER	295	265
ADVANTAGE GRAPHICS	250	225
ADVANTAGE LINK	395	359
ADVANTAGE MAKE	125	99
PANEL PLUS	495	395
QUICKSCREEN	295	265
RUN/C PROFESSIONAL	250	155
TIMESLICER	295	265
W/SOURCE	1000	CALL
MICROSOFT		
BUS MOUSE	175	129
C COMPILER	450	269
COBOL	700	429
FORTRAN	450	269
LISP	250	159
MACRO ASSEMBLER	150	93
MUMATH	300	185
PASCAL	300	185
QUICKBASIC	99	65
SORT	195	125
WINDOWS DEVELOPMENT KIT	500	309
PHOENIX		
PASM 86	195	109
PDISK	145	89
PFANTASY PAC	995	599
PFINISH	395	229
PFIX 86 PLUS	395	229
PFORCE	395	209
PFORCE++	395	215
PLINK 86 PLUS	495	279
PMAKER	125	79
PMATE	195	115
PRE-C	295	155
PTEL	195	115
POLYTRON		
C BEAUTIFIER	49	45
C LIBRARY I	99	79
COUNTRY CODE	679	CALL
POLYBOOST	80	69
POLYDESK III	99	79
POLYDESK III ADD ONS	CALL	CALL
POLYLIBRARIAN	99	79
POLYLIBRARIAN II	149	115
POLYMAKE 2.0	149	119
POLYSHELL	149	115
POLYXREF	219	169
POLYXREF— ONE LANGUAGE	129	99
POWERCOM	179	139
PVCS PERSONAL	149	115
PVCS CORPORATE	395	309
PVCS NETWORK	1000	CALL
OPERATING SYSTEMS		
OPERATING SYSTEM		
WENDIN TOOLBOX	99	79
PCNX	99	79
PCVMS	99	79
MICROPORT SYSTEM V/AT	549	465
DOS MERGE 286	149	125
RUNTIME SYSTEM	199	169
SOFTWARE DEV. SYSTEM	249	209
TEXT PREPARATION SYSTEM	199	169
UNLIMITED USER UPGRADE	249	209
SCO XENIX SYSTEM V	1295	995
XENIX DEV. SYSTEM	595	499
XENIX OPERATING SYSTEM	595	499
XENIX TEXT PROC. PKG.	195	145
DESKTOP PUBLISHING		
HALO DPE	195	155
XEROX VENTURA PUBLISHER	895	695
TURBO VENTURA PUBLISHER	1595	1249

XENIX/UNIX PRODUCTS

	LIST	OURS
ADVANTAGE C++	695	CALL
BTRIEVE	595	459
C-ISAM	319	285
C-TERP	498	379
C-TREE	395	329
DB_VISTA	CALL	CALL
FOXBASE	795	CALL
INFORMIX ESQL/C	749	599
INFORMIX 4GL	1500	1235
INFORMIX SQL	995	795
LYRIX	595	449
MICROSOFT BASIC COMPILER	695	439
MICROSOFT BASIC INTERPRETER	350	229
MICROSOFT COBOL	995	629
MICROSOFT FORTRAN	695	429
MICROSOFT PASCAL	695	429
NETWORKS FOR XENIX	595	495
PANEL	625	535
RM/COBOL	1250	949
RM/FORTRAN	750	549
SCO PROFESSIONAL	795	595
UNIWARE 68020 C CROSS	1390	CALL

	LIST	OURS
TELECOMMUNICATIONS		
ASCOM IV	195	175
PTEL	195	115
SIDETALK	120	89
ADDITIONAL PRODUCTS		
C TOOL SET	95	CALL
COMMAND PLUS	80	69
DAN BRICKLIN'S DEMO PROGRAM	75	59
DISK OPTIMIZER	60	55
FASTBACK	175	135
HELP/CONTROL	125	99
INTERACTIVE EASYFLOW	150	129
LATTICE TEXT MGMT. UTIL.	120	89
MICROSOFT MACH 10 W/MOUSE & WINDOWS	549	379
MKS TOOLKIT	139	109
NORTON COMMANDER	75	55
NORTON UTILITIES	100	59
PC/EDT	250	CALL
PDISK	145	89
SAPIENS V8	300	269
SCREEN MACHINE	79	59
SCROLL & RECALL	69	59
SECURITY LIBRARY	125	115
SOURCE PRINT	75	59
TASKVIEW	80	59
TLIB	100	89
TREE DIAGRAMMER	55	50
ZAP	95	CALL

FEATURED PRODUCT OF THE MONTH

Microport System V/AT— New release 2.2 of complete UNIX system has numerous enhancements, faster speed, improved documentation. Includes free Nutshell guide to UNIX and Sysvision, a user-friendly system administrator front end to UNIX.
List $549 **Ours $465**

CALL TOLL FREE: **1-800-445-7899**
In NY: 914-332-4548

Terms and Policies

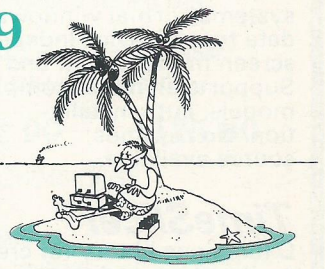
- **We honor MC, VISA, AMERICAN EXPRESS**

No surcharge on credit card or C.O.D. Prepayment by check. New York State residents add applicable sales tax. Shipping and handling $3.00 per item, sent UPS ground. Rush service available, prevailing rates.

- Programmer's Paradise will match any current nationally advertised price for the products listed in this ad.
- **Mention this ad when ordering**— some items are specially priced.
- Prices and Policies subject to change without notice.
- Corporate and Dealer inquiries welcome.

*Ask for details. Some manufacturers will not allow returns once disk seals are broken.

Programmer's Paradise



42 River Street, Tarrytown, NY 10591

DISCOVER PARADISE CONTEST

Win The High-Performance Software Development Package, Valued At Over $2,000!

Now that you've seen what Programmer's Paradise looks like, we want you to guess where it is! Just search this ad for clues to the identity of our tropical island paradise (Hint: They're mixed in with our product listings). Then send in your guess on this coupon. You could be the lucky winner of the high-performance software development package**, featuring: • **Univation's Dream Board**—Multifunction accelerator board with EMS. Designed to fit into a single expansion slot. The Dream Board combines the functions of three of the most popular enhancement boards—multifunction, EMS, accelerator—into one power-packed package • **ADVANTAGE Link** overlay linker • **Microsoft** compiler of your choice (MS/PC-DOS) • **Editor** of your choice (binary). **... VALUE: OVER $2,000**

UNIVATION System Solutions For Business

LIFEBOAT Microsoft®

Complete this form and mail to: DISCOVER PARADISE CONTEST, c/o Programmer's Paradise, 42 River Street, Tarrytown, NY 10591. Entries must be received by August 1, 1987.

OFFICIAL RULES: 1. Limit one name per entry, no purchase necessary. Enter as often as you wish, mailing each entry on a separate postcard. All entries must be received by August 1, 1987. **2.** Participants must be 18 years or older and a resident of the United States. Void where prohibited by law. **3.** Winner will be responsible for all taxes. **4.** Odds of winning will be determined by number of entries received. Winner will be selected out of all correct entries received. **5.** Prize will be awarded within 30 days of all entries received.
**No substitutions allowed.

I think the name of your island is: _____

Name _____ **Title** _____

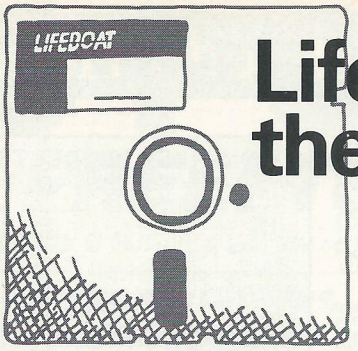
Company name _____

Company address _____

City _____ **State** _____ **Zip** _____

Phone # _____ 7CL

CIRCLE 27 ON READER SERVICE CARD



Lifeboat. Your best source for the best names in software.

Microsoft® COBOL

New v. 2.2 COBOL compiler now includes Microsoft COBOL TOOLS, COBOL cross-reference generator and ViewCob interactive symbolic debugger.

Microsoft

C86PLUS

Proprietary C compiler design applies artificial intelligence to produce highly optimized code. Provides a highly productive and sophisticated programming environment, especially in ROM applications development. Features Microsoft C v. 4.0 and UNIX V compatibility; C library source code; full ANSI libraries and over 300 functions; ROMable code; small, compact, medium and large memory model support; 186/286/386 code generations options; and 8087/287 math support and auto detect emulator.

C86PLUS™

Greenleaf Data Windows

New concept C library features overlayed logical windows, transaction data entry, multiple menu systems, virtual windows, many data types, device independence, screen management and more. Supports all major compilers, all models, auto installation. No royalties; source available.

GREENLEAF Software

TimeSlicer™

C functions library for creating multitasking and real-time programs at the application level. Create, suspend or terminate tasks at run-time. Compatible with Lattice C, Microsoft C, ADVANTAGE C++ and object-oriented programming.

Latest From LIFEBOAT

Call for the latest 80386 development software.

ADVANTAGE C++™

Expand your programming capabilities with C++, the object-oriented language that gives you all the benefits of C without its limitations. Our enhanced ADVANTAGE C++ is the only full C++ implementation available. Develop large and complex programs with greater resilience, fewer bugs. Code is more reliable and maintainable. Fully tested and documented; compatible with existing C programs and tools. Available for Lattice and Microsoft C compilers; MS-DOS and XENIX operating systems.

Latest From LIFEBOAT

VEDIT PLUS

For years, thousands of programmers have depended on VEDIT. If you take your editing seriously, take a good look at the all-new VEDIT PLUS. Open windows to simultaneously edit several files. Access editing functions with pop-up menus. Use keystroke macros to speed editing. And run other programs within VEDIT PLUS—all with uncompromising speed, flexibility and power. VEDIT PLUS is completely customizable and available for MS/PC-DOS, CP/M 80/86. Ask for your FREE demo/tutorial disk.

CompuView

Microsoft® QuickBASIC

New v. 3.0 high-performance BASIC compiler offers BASICA compatibility plus many enhancements for faster programming: Improved integrated debugging; increased structured and modular programming support; greatly improved math support including in-line 8087/287 code generation; advanced integrated editor; dramatically faster execution speed.

Microsoft

Periscope III

Periscope III sets a new standard in price/performance for real-time hardware breakpoint debuggers. You'll find the errors in real-time systems, stop intermittent failures, interface with undocumented systems and eliminate bottlenecks in your code. It's an easy transition from other models too, since the commands are a superset of those used in Periscope I, II and II-X. One board works on PC/XT/AT. The system includes the board, breakout switch, software, manual and quick reference card.

PERISCOPE

Pascal-2

Have you reached the limits of Turbo? Upgrade to Pascal-2 without loss. Easy migration path from Turbo with compatible strings, equivalent procedures and access to Turbo graphics. Cut execution time by 200% over Turbo. Use all of DOS-accessible memory through efficient, large memory model. Speed development time with sophisticated error checking and reporting. Call Microsoft languages. Plus interactive source-level debugger, error walkback, Intel CEL87 math library, high-level profiler.

OREGON SOFTWARE

Call today for your FREE Scientific & Engineering Solutions catalog!

Describes 100 of the best software packages for solving complex equations, number crunching, analyzing data and more.

Call **1-800-847-7078**
In NY **914-332-1875**
or see your local Lifeboat Affiliated Dealer

The Full-Service Source for Programming Software

LIFEBOAT

55 South Broadway
Tarrytown, NY 10591
Telex # 510-610-7602

The names of the products listed are generally the trademarks of the sources of the products.

INTERNATIONAL SALES OFFICES

Australia/New Zealand: MoS Computer Software/Charlton Distributors Auckland (09) 766-361
Canada: Scantel Systems Toronto (416) 449-9252
Denmark: Ravenholm Copenhagen 288-7249
England: Grey Matter, Ltd. Devon 364-53499
System Science, Ltd. London (01) 248-0962
France: Compusol Paris 14 530 0737
Italy: Lifeboat Associates Italia Milan 02-464601
Japan: Lifeboat, Inc. Tokyo 03-293-4711
SATT Software Tokyo 03-295-3390
Netherlands: SCOS Automation BV Amsterdam 020-10 69 22
Spain: Micronet, S.A. Madrid 1-262-3304
Switzerland: Euro-Link Zurich 813 514
West Germany: MEMA Computer GmbH Frankfurt 069-347226
Omnitex Rheinfelden 07623/61820

Object-Oriented Programming with SCOOPS

Implementing the scissors-paper-stone game with this facility of LISP-based PC Scheme shows off the benefits of object-oriented programming

By John D. McGregor

In his column "Scissors, paper, stone: A tournament of Schemes,"[1] Brian Hayes developed an interesting implementation of this popular child's game using a dialect of LISP from Texas Instruments called PC Scheme.[2]

In addition to presenting the special features of Scheme, a central theme of Hayes's column was the power of returning procedures as the result of a function. This approach was taken to provide a degree of data abstraction as well as a method for saving the state of a set of variables.

After reviewing the implementation and the reasons for using a procedure to return a procedure, I was struck by the similarity of this technique with an object-oriented approach. Since I happened to be looking for a simple object-oriented example to use in a class, I immediately went to work on a sample system. PC Scheme includes an object-oriented facility, the Scheme Object-Oriented Programming System (SCOOPS), and thus was a natural choice for the implementation.

This article gives a brief introduction to object-oriented programming in general and the features of SCOOPS in particular. The basic features of this implementation follow those in the Hayes article to aid you in comparing the two approaches. Before discussing objects, let's review the game.

The simplicity of this game makes it a good example. The cyclic arrangement of values (Figure 1) means there is no sure offense, but it is possible to construct a variety of defenses and strategies.

In this game, two players each choose a move from the three possible. The winner is decided based on the relationships shown in Figure 1. One point is scored for every match a player wins. An implementation of this game should include representations for the two players, with appropriate procedures for generating moves.

In his discussion, Hayes was concerned with the privacy of the score. He didn't want players that could alter the score to be implemented. Thus he created a separate procedure to protect the score. In my implementation, an object (the Referee) provides this protection. Before describing in detail this object-oriented implementation, an introduction to objects is in order.

Objects

The designer using object-orientation seeks to identify the major entities in the developing system. Objects are developed to describe each of the entities. An **object** is a collection of data and procedures that operate on that data. The data of the object represents a private memory for the object. The procedures associated with the object are the only legitimate operators for that data.

Data base systems, expert systems, and other types of systems are benefiting from the object-oriented approach. One reason for the growing attention to this technique is its support of data encapsulation. The data in an object can only be acted upon by the procedures contained within the object. An outside entity informs the object what it wants done, such as displaying a data value, but it does not tell the object how to carry out the task. The exact details of the data structures are hidden from the outside entity.

A second reason for the growing importance of object-oriented programming is the ability it provides to save the local state. In a procedural language, local variables do not retain their values between invocations of the procedure. While global variables can be used to retain values, the data is no longer within the protective boundaries of a single procedure. Although the variables of an object behave like local procedural variables in some ways, they can retain their values between messages to the object.

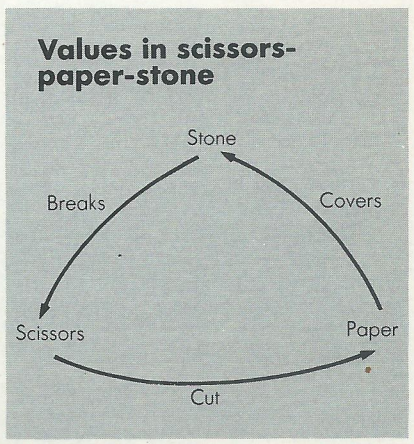


Figure 1.

Classes and instances. In PC Scheme, an object is implemented by first defining a class. A **class** can be thought of as a template for a category of objects. This basic definition reserves a slot for each of the data items that make up the local memory of the object. Consider a possible SCOOPS definition for the object class representing a computer science faculty member:

```
(define-class cs-faculty-member
      (classvars dept-chair)
      (instvars name ss-number specialty))
```

This statement creates a class with four variables, but it is only a definition. This abstraction must be instantiated before memory is allocated to store values. An instance of this class would be created by a statement like:

```
(define JOHN (make-instance
   cs-faculty-member
         'dept-chair    'BILL
         'name          "John McGregor"
         'ss-number     '000-00-0000
         'speciality    'AI          ))
```

The class variable *dept-chair* is global to the *cs-faculty-member* class. Thus each instance sees the same value for the *dept-chair* variable, has access to this value, and can modify it. Any modification of a class variable is immediately available to all instances of that class.

The values of the instance variables, name, Social Security number, and specialty are local to their particular instance. Their values can only be altered by the methods associated with that instance. Even though these variables are local, their values are retained between operations on the instance.

Methods and messages. The procedures associated with an object are termed methods. A **method** acts on the data associated with its object. A method to change a faculty member's specialty might look like:

```
(define-method (cs-faculty-member
   change-specialty)(new-value)
      (set! specialty new-value))
```

A method is activated by sending a message to the instance that is to be the receiver of the action. A message is made up of the keyword *SEND,* the identifier for a particular instance, a selector that is the name of one of the object's methods, and any arguments needed for the method. A message to activate the method in the previous example would be stated as:

```
(send JOHN change-specialty
   "Graphics")
```

An important and much-debated concept in object-oriented systems is early vs. late bindings. Delaying binding to the time of message passing allows a number of classes to possess the same message selectors, which are implemented internally to the objects in different ways, providing a higher degree of abstraction.

Language implementations such as SCOOPS and Objective-C provide for late bindings and thus for developing a set of abstract operations common across many classes. (See the accompanying sidebar, "Object-oriented implementations," for more on object-oriented languages.)

An object definition creates a template from which specific instances are defined. An object includes both local data storage and a set of associated methods for manipulating that data. Instance variables retain their values between messages to the object. If methods do not exist to manipulate certain of the object's variables, those values cannot be changed by outside agents.

```
;CLASS GAME
;THIS CLASS IS A SHELL THAT INSTANTIATES A REFEREE INSTANCE TO CONTROL
;THE GAME.  IT THEN SIGNALS THE REFEREE INSTANCE TO BEGIN THE GAME
;
(DEFINE-CLASS GAME
    (INSTVARS (WINNING-SCORE 15) THE-REFEREE INSTRUCTIONS)
    (OPTIONS GETTABLE-VARIABLES SETTABLE-VARIABLES INITTABLE-VARIABLES))
(DEFINE-METHOD (GAME RESET-WINNING-SCORE)()
    (BEGIN (DISPLAY "ENTER VALUE FOR WINNING SCORE ")
           (SET! WINNING-SCORE (READ))))
(DEFINE-METHOD (GAME START)()
    (BEGIN
           (DISPLAY "DO YOU WANT TO SET GAME VALUES? (Y/N) ")
           (SET! INSTRUCTIONS (READ-CHAR))
           (COND ((OR (EQV? INSTRUCTIONS #\y) (EQV? INSTRUCTIONS #\Y))
                 (RESET-WINNING-SCORE)))
           (SET! THE-REFEREE (MAKE-INSTANCE REFEREE))
           (SEND THE-REFEREE START-GAME)))
;
;CLASS REFEREE
;THE CURRENT INSTANCE OF REFEREE SIGNALS EACH PLAYER TO MAKE A MOVE.
;IT DETERMINES THE WINNER OF EACH MATCH, KEEPS SCORE, AND
;TERMINATES A GAME WHEN THE WINNING SCORE IS REACHED
;
(DEFINE-CLASS REFEREE
    (INSTVARS (SCORE1 0) (SCORE2 0) MOVE-PLAYER-1 MOVE-PLAYER-2
    PLAYER-1 PLAYER-2 )
    (OPTIONS GETTABLE-VARIABLES SETTABLE-VARIABLES INITTABLE-VARIABLES))
(DEFINE-METHOD (REFEREE START-GAME) ()
    (BEGIN
           (SET! PLAYER-1 (MAKE-INSTANCE PLAYER))
           (SET! PLAYER-2 (MAKE-INSTANCE PLAYER))
           (PLAY)
           (WRITELN "GAME COMPLETED")))
(DEFINE-METHOD (REFEREE PLAY) ()
    (BEGIN (DO ((I 1 (1+ I)))((OR (=? SCORE1 WINNING-SCORE)(=? SCORE2
                                                    WINNING-SCORE)))
```

Listing 1. *(Continued on following page)*

First implementation

A simple implementation of the scissors-paper-stone game is presented in Listing 1. The game is represented using three object classes: *Game, Referee,* and *Player.* The attributes (instance variables) for each class are shown in Figure 2.

The top-level *Run* function begins the game by generating an instance of class *Game* identified as *THIS-GAME.*

Game. The *Game* object is the overall manager of the simulation. The instance of the *Game* object, which is produced by *Run,* is responsible for generating an instance of the *Referee* object. *Game* also gives the user the opportunity to set the size of the winning score. *THIS-GAME* then signals the *Referee* instance to begin play by sending the message "START-GAME."

PC Scheme does not support class definitions as objects so it is not possible to generate instances by sending a message to the class definition, as one might

```
      (SET! MOVE-PLAYER-1 (SEND PLAYER-1 MAKE-MOVE))
      (SET! MOVE-PLAYER-2 (SEND PLAYER-2 MAKE-MOVE))
      (WRITELN "    " MOVE-PLAYER-1 #\TAB MOVE-PLAYER-2)
      (SEND PLAYER-1 TRACK-MOVE MOVE-PLAYER-2)
      (SEND PLAYER-2 TRACK-MOVE MOVE-PLAYER-1)
      (COND ((OR (AND (EQUAL? MOVE-PLAYER-2 'SCISSORS)
                      (EQUAL? MOVE-PLAYER-1 'PAPER))
                 (AND (EQUAL? MOVE-PLAYER-2 'PAPER)
                      (EQUAL? MOVE-PLAYER-1 'STONE))
                 (AND (EQUAL? MOVE-PLAYER-2 'STONE)
                      (EQUAL? MOVE-PLAYER-1 'SCISSORS)))
                 (SET! SCORE2 (1+ SCORE2)))
                 ((OR (AND (EQUAL? MOVE-PLAYER-1 'SCISSORS)
                           (EQUAL? MOVE-PLAYER-2 'PAPER))
                      (AND (EQUAL? MOVE-PLAYER-1 'PAPER)
                           (EQUAL? MOVE-PLAYER-2 'STONE))
                      (AND (EQUAL? MOVE-PLAYER-1 'STONE)
                           (EQUAL? MOVE-PLAYER-2 'SCISSORS)))
                      (SET! SCORE1 (1+ SCORE1)))
                 (ELSE
                      (WRITELN "DRAW ON THIS MATCH")))
            (COND ((=? SCORE1 WINNING-SCORE)
                      (NEWLINE )(DISPLAY "  PLAYER-1 WON"))
                  ((=? SCORE2 WINNING-SCORE)
                      (NEWLINE)(DISPLAY "  PLAYER-2 WON"))))))
;
;CLASS PLAYER
;TWO INSTANCES OF PLAYER ARE INSTANTIATED FOR EACH GAME
;
(DEFINE-CLASS PLAYER
    (INSTVARS  (OP-1 0)(OP-2 0) (OP-3 0) CURRENT-MOVE
               (MOVES (VECTOR 'STONE 'PAPER 'SCISSORS)))
    (OPTIONS GETTABLE-VARIABLES SETTABLE-VARIABLES INITTABLE-VARIABLES))
(DEFINE-METHOD (PLAYER MAKE-MOVE)()
    (BEGIN (SET! CURRENT-MOVE  (VECTOR-REF MOVES (RANDOM 3)))
           (CAR (LIST CURRENT-MOVE))))
(DEFINE-METHOD (PLAYER TRACK-MOVE)(VAL)
    (BEGIN (COND ((EQUAL? VAL 'STONE) (SET! OP-1 (1+ OP-1)))
                 ((EQUAL? VAL 'PAPER) (SET! OP-2 (1+ OP-2)))
                 ((EQUAL? VAL 'SCISSORS) (SET! OP-3 (1+ OP-3))))))
(DEFINE RUN
    (LAMBDA () (DEFINE THIS-GAME (MAKE-INSTANCE GAME))
           (SEND THIS-GAME START)))
```

Listing 1. *(Continued from preceding page)*

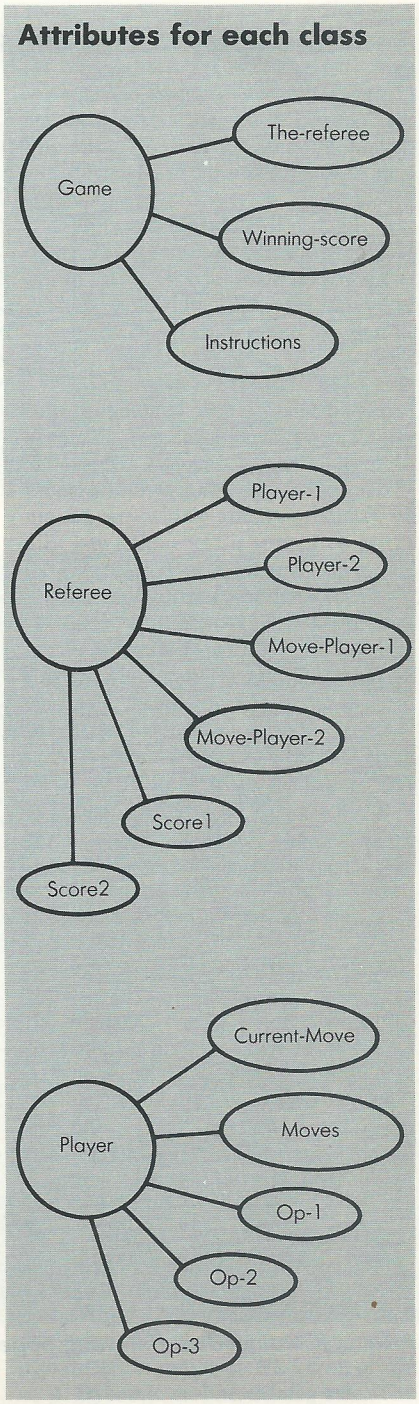


Figure 2.

Relationship of classes and instances for simulation

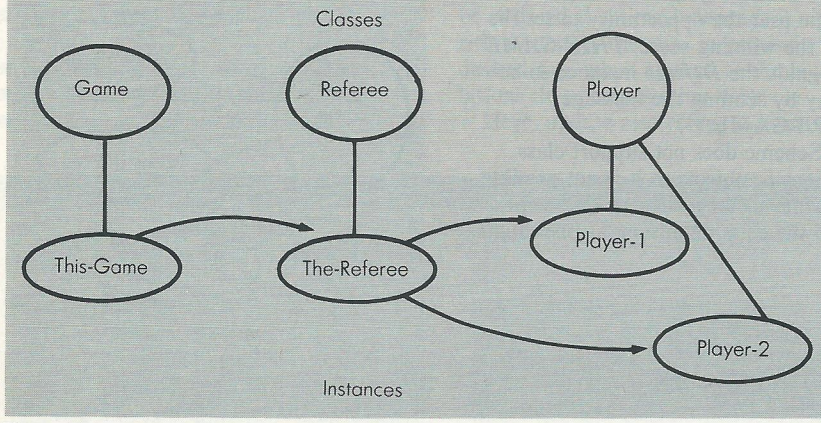


Figure 3.

```
; class Window
; Window class displays output to screen
; This class uses pop-up windows, and LISP handles overlap
; automatically
;
(define-class window
   (instvars x-origin y-origin x-size y-size title name)
   (options gettable-variables settable-variables inittable-variables))
(define-method (window create) ()
   (begin (set! name (make-window title #!true))
          (window-set-size! name y-size x-size)
          (window-set-position! name y-origin x-origin)))
(define-method (window move) (new-x new-y)
   (begin (window-popup-delete name)
          (window-set-position! name new-y new-x)
          (window-popup name)))
(define-method (window show) ()
   (window-popup name))
(define-method (window erase) ()
   (window-popup-delete name))
;
;CLASS GAME
;THIS CLASS IS A SHELL THAT INSTANTIATES A REFEREE INSTANCE TO CONTROL
;THE GAME.  IT THEN SIGNALS THE REFEREE INSTANCE TO BEGIN THE GAME
;
(DEFINE-CLASS GAME
    (INSTVARS (WINNING-SCORE 15) THE-REFEREE INSTRUCTIONS)
    (MIXINS WINDOW)
    (OPTIONS GETTABLE-VARIABLES SETTABLE-VARIABLES INITTABLE-VARIABLES))
(DEFINE-METHOD (GAME RESET-WINNING-SCORE)()
    (BEGIN (NEWLINE NAME)
           (DISPLAY "ENTER VALUE FOR WINNING SCORE " NAME)
           (SET! WINNING-SCORE (READ NAME))))
```

Listing 2. *(Continued on following page)*

in Smalltalk. Figure 3 shows the relationship of the classes and instances for the overall simulation. The arrows illustrate the direction of messages.

Referee. The referee is the central controller of the game. The *Referee* object makes two instances of the *Player* class. After setting up a table for showing game moves, the *Referee* instance invokes its *PLAY* method to begin a game.

Note that in many object-oriented implementations, the variable *self* is included in each new class by inheritance from a root *Object* class. This variable is used by an object to send a message to itself to invoke another method. PC Scheme allows one method to directly invoke another method for the same object.

The referee signals each player when a new move is needed by sending them a "MAKE-MOVE" message. Once both players have responded with new moves, the referee determines the winner (if there is one) of each match by examining the contents of the *MOVE-PLAYER-1* and *MOVE-PLAYER-2* variables.

The referee also maintains the score for each player and declares a winner. Since no method allows an outside agent to change the score, it is protected. The referee also notifies each player of the move made by the other player.

Player. Each player is responsible for generating moves when they are requested by the referee. In this implementation, the players simply use a randomly generated value for one of the three moves. The players also record their opponent's moves but make no use of the information at present.

Inheritance

Object-oriented systems handle complexity by building a system in layers. New classes gain many of their capabilities by inheriting the capabilities of previously defined, usually more general, objects.

Windows. The previous implementation of SCOOPS did a very poor job of output. In Listing 2, the new object class *Window* is added. This class provides a set of variables and methods for creating and displaying windows. Each of the existing objects has been modified to inherit it from the window class by adding the statement *(mixins window)* to the definition for each of the classes. This state-

```
;CLASS REFEREE
;THE CURRENT INSTANCE OF REFEREE SIGNALS EACH PLAYER TO MAKE A MOVE.
;IT DETERMINES THE WINNER OF EACH MATCH, KEEPS SCORE, AND
;TERMINATES A GAME WHEN THE WINNING SCORE IS REACHED
;
(DEFINE-CLASS REFEREE
    (INSTVARS (SCORE1 0) (SCORE2 0) MOVE-PLAYER-1 MOVE-PLAYER-2
    PLAYER-1 PLAYER-2 X)
    (MIXINS WINDOW)
    (OPTIONS GETTABLE-VARIABLES SETTABLE-VARIABLES INITTABLE-VARIABLES))
(DEFINE-METHOD (REFEREE START-GAME) ()
    (BEGIN
          (SET! X-ORIGIN 10) (SET! Y-ORIGIN 4)
          (SET! X-SIZE   45) (SET! Y-SIZE   8)
          (SET! TITLE "REFEREE")
          (CREATE) (SHOW)
          (SET! PLAYER-1 (MAKE-INSTANCE PLAYER
                              'X-ORIGIN 5
                              'TITLE    "PLAYER - 1"))
          (SET! PLAYER-2 (MAKE-INSTANCE PLAYER
                              'X-ORIGIN 40
                              'TITLE    "PLAYER - 2"))
          (PLAY)
          (WRITELN "GAME COMPLETED")))
(DEFINE-METHOD (REFEREE PLAY) ()
    (BEGIN (DO ((I 1 (1+ I)))((OR (=? SCORE1 WINNING-SCORE)
                                  (=? SCORE2 WINNING-SCORE)))
          (SET! MOVE-PLAYER-1 (SEND PLAYER-1 MAKE-MOVE))
          (SET! MOVE-PLAYER-2 (SEND PLAYER-2 MAKE-MOVE))
          (SEND PLAYER-1 SHOW-MOVE)
          (SEND PLAYER-2 SHOW-MOVE)
          (SEND PLAYER-1 TRACK-MOVE MOVE-PLAYER-2)
          (SEND PLAYER-2 TRACK-MOVE MOVE-PLAYER-1)
          (COND ((OR (AND (EQUAL? MOVE-PLAYER-2 'SCISSORS)
                          (EQUAL? MOVE-PLAYER-1 'PAPER))
                     (AND (EQUAL? MOVE-PLAYER-2 'PAPER)
                          (EQUAL? MOVE-PLAYER-1 'STONE))
                     (AND (EQUAL? MOVE-PLAYER-2 'STONE)
                          (EQUAL? MOVE-PLAYER-1 'SCISSORS)))
                 (SET! SCORE2 (1+ SCORE2))
                 (SHOW) (SHOW-SCORE))
                ((OR (AND (EQUAL? MOVE-PLAYER-1 'SCISSORS)
                          (EQUAL? MOVE-PLAYER-2 'PAPER))
                     (AND (EQUAL? MOVE-PLAYER-1 'PAPER)
                          (EQUAL? MOVE-PLAYER-2 'STONE))
                     (AND (EQUAL? MOVE-PLAYER-1 'STONE)
                          (EQUAL? MOVE-PLAYER-2 'SCISSORS)))
                 (SET! SCORE1 (1+ SCORE1))
                 (SHOW) (SHOW-SCORE))
                (ELSE
                   (SHOW) (SHOW-SCORE)
                   (DISPLAY "    DRAW ON THIS MATCH" NAME)))
          (COND ((=? SCORE1 WINNING-SCORE)
                 (DISPLAY "    PLAYER-1 WON" NAME))
                ((=? SCORE2 WINNING-SCORE)

(DEFINE-METHOD (GAME START)()
    (BEGIN
          (SET! X-ORIGIN 1) (SET! Y-ORIGIN 1)
          (SET! X-SIZE 68)  (SET! Y-SIZE   20)
          (SET! TITLE "STONE PAPER SCISSORS GAME")
          (CREATE)
          (SHOW)
          (DISPLAY "DO YOU WANT TO SET GAME VALUES? (Y/N) " NAME)
          (SET! INSTRUCTIONS (READ-CHAR NAME))
          (COND ((OR (EQV? INSTRUCTIONS #\y) (EQV? INSTRUCTIONS #\Y))
                 (RESET-WINNING-SCORE)))
          (ERASE)
          (SHOW)
          (SET! THE-REFEREE (MAKE-INSTANCE REFEREE))
          (SEND THE-REFEREE START-GAME)))

                   (DISPLAY "    PLAYER-2 WON" NAME)))
          (DO ((I 1 (1+ I)))          ;DELAY LOOP
              ((> I 7000) I)
              (SET! X 0))
          (ERASE)
          (SEND PLAYER-1 ERASE)
          (SEND PLAYER-2 ERASE))))
```

Listing 2. *(Continued on following page)*

DAN BRICKLIN'S DEMO PROGRAM ONLY $74.95

Read what they're saying about this popular program for prototyping and demo-making:

"A winner right out of the starting gate. After you use DEMO once, you'll wonder how you got along without it."
—PC Magazine

"Everybody who writes software, either commercially or for in-house applications, should immediately order a copy. Period. No exceptions."
—Soft·letter

Product of the Month
—PC Tech Journal

Thousands of developers and most of the largest and best known software companies are using this program. You can, too. Act now!

NEW TUTORIAL! JUST $49.95

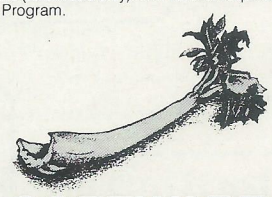
The perfect companion to the Demo Program. The Tutorial helps you learn the ins and outs of its basic and advanced features. Complete with a 96 page manual containing step-by-step instructions, diskette, and function key template.

ORDER NOW!

1-800-CALL-800 x8088

VISA MasterCard AMERICAN EXPRESS

Use 800-number for orders only.
Questions, special shipping, etc., call **617-332-2240.**
No Purchase Orders. Massachusetts residents add 5% sales tax. Outside of the U.S.A., add $15.00.
Requires 256K IBM PC/Compatible, DOS 2.0 or later. Supports Monochrome, Color Graphics, and EGA Adapters (text mode only). The Tutorial requires the Demo Program.



SOFTWARE GARDEN, INC.
Dept. L
P.O. Box 373, Newton Highlands, MA 02161

CIRCLE 29 ON READER SERVICE CARD

```
(DEFINE-METHOD (REFEREE SHOW-SCORE)())
    (BEGIN (NEWLINE NAME)(NEWLINE NAME)
           (DISPLAY "          " NAME)
           (DISPLAY "Player 1" NAME)
           (DISPLAY "          " NAME)
           (DISPLAY "Player 2" NAME)
           (NEWLINE name)
           (DISPLAY "          " NAME)
           (DISPLAY SCORE1 NAME)
           (DISPLAY "                " NAME)
           (DISPLAY SCORE2 NAME)))
;
;CLASS PLAYER
;TWO INSTANCES OF PLAYER ARE INSTANTIATED FOR EACH GAME
;
(DEFINE-CLASS PLAYER
    (INSTVARS  (OP-1 0)(OP-2 0) (OP-3 0) CURRENT-MOVE
               (Y-ORIGIN 15) (X-SIZE 12) (Y-SIZE 4)
               (MOVES (VECTOR 'STONE 'PAPER 'SCISSORS)))
    (MIXINS WINDOW)
    (OPTIONS GETTABLE-VARIABLES SETTABLE-VARIABLES INITTABLE-VARIABLES))
(DEFINE-METHOD (PLAYER MAKE-MOVE)()
    (BEGIN (SET! CURRENT-MOVE  (VECTOR-REF MOVES (RANDOM 3)))
           (CAR (LIST CURRENT-MOVE))))
(DEFINE-METHOD (PLAYER TRACK-MOVE)(VAL)
    (BEGIN (COND ((EQUAL? VAL 'STONE) (SET! OP-1 (1+ OP-1)))
                 ((EQUAL? VAL 'PAPER) (SET! OP-2 (1+ OP-2)))
                 ((EQUAL? VAL 'SCISSORS) (SET! OP-3 (1+ OP-3))))))
(define-method (player show-move) ()
    (begin (CREATE) (show)
           (newline name)
           (newline name)
           (DISPLAY "  " NAME)
           (display CURRENT-MOVE name)))
(DEFINE RUN
    (LAMBDA () (DEFINE THIS-GAME (MAKE-INSTANCE GAME))
           (SEND THIS-GAME START)))
```

Listing 2. ***(Continued from preceding page)***

Diagramming technique

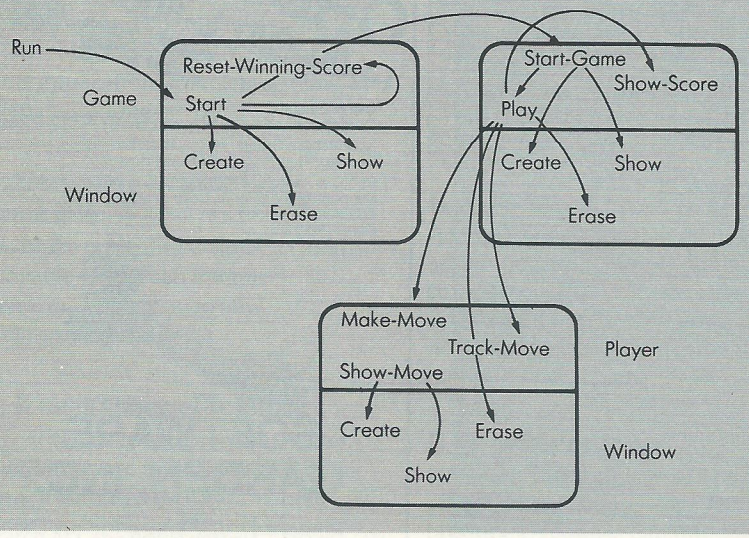


Figure 4.

ment defines, for example, a method *create* for the *Game, Referee,* and *Player* objects that allows instances of these objects to create windows for output purposes.

Inheritance means that the methods and variables defined in the superclass become part of the inheriting class. The instance variables defined in the window class become instance variables for each instance of the classes. Thus the inheriting class gains both capabilities and information without the class definition becoming more complex.

Figure 4 uses a diagramming technique proposed by W. Cunningham and K. Beck[3] to illustrate the layering of classes and interaction through messages. Note that superclasses are beneath the inheriting class, thus emphasizing that the appropriate method is found by a depth-first search.

Strategic decisions. As an additional example of inheritance, Listing 3 contains two additional class definitions: *ODDS-PLAYER* and *AVERAGE-PLAYER*. These two classes provide definitions of players that use a more sophisticated (though not by much) strategy for generating moves. *ODDS-PLAYER* selects as its next move the move most often used by its opponent, while *AVERAGE-PLAYER* selects the move least often used by its opponent.

These new classes illustrate the power of inheritance. Most of the variables and methods needed are obtained from the superclass *Player*. The *MAKE-MOVE* method defined in the new definitions has the same name and purpose as one of the original methods in *Player*. When a message invokes a method, a depth-first search begins at the class that defines the instance object and searches through the inheritance chain to find the appropriate code.

When an *ODDS-PLAYER* instance receives the message "ERASE," a search begins at the *ODDS-PLAYER* definition and continues down the chain until the appropriate code is found in the *WINDOW* definition. If the message were "MAKE-MOVE," the search would stop with *ODDS-PLAYER*, which has a *MAKE-MOVE* method defined and would not find the *MAKE-MOVE* meth-

od defined in *PLAYER*. Thus several method definitions may share the same name, but only the most recent definition will be used.

Comparing implementations

The concept of object-oriented programming is too recent to be highly standardized. G. Pascoe[4] gives four fundamental characteristics of object orientation: information hiding, data abstraction, dynamic binding, and inheritance.

SCOOPS supports information hiding by invoking methods through messages. The implementation of the method is hidden from the message sender. Data abstraction is achieved by the overloading of method selectors. Since SCOOPS is a LISP-based implementation, the usual dynamic binding process is utilized. Finally, as already discussed, SCOOPS supports inheritance of methods and variables using the *mixins* clause of the class definition statement.

Additional features are present in other languages that claim to be object-oriented. PC Scheme's SCOOPS facility is a useful object-oriented facility, but it is not complete when compared to some of the other available implementations.

What's missing. The differences begin with creating a new instance of an object. Objective-C and Smalltalk allow sending messages to a class definition. For example, if a Smalltalk image includes a definition for a *Frame* object, a new instance of *Frame* is created by the message "Frame new." Although this approach differs little from SCOOPS' *MAKE-INSTANCE*, a message approach would be more consistent.

To present a consistent interface, many object-oriented languages such as Smalltalk include the special instance variable *self* (as already mentioned), which is used by an object to send messages to itself. This ability allows all procedure activations to occur through messages.

SCOOPS does not have this variable, but object methods can be invoked as ordinary procedures by other methods within the object. In the *START-GAME* method of the *Referee* object, it would be more consistent to use *(Send self Play)* as opposed to *(Play)*.

The pluses. PC Scheme and several other implementations support a form of polymorphism. Scheme implements ad hoc polymorphism through function name overloading, which allows different objects to present the same interface to the outside world by having them all respond to the same messages. In Listing 3, for example, *ODDS-PLAYER* and *AVERAGE-PLAYER* instances each respond to "MAKE-MOVE" messages but use somewhat different algorithms.

Multiple inheritance is available in most LISP-based, object-oriented systems but not in C++ or most commercial versions of Smalltalk (with the

```
;CLASS PLAYER
;TWO INSTANCES OF PLAYER ARE INSTANIATED FOR EACH GAME
;
(DEFINE-CLASS PLAYER
    (INSTVARS  (OP-1 Ø)(OP-2 Ø) (OP-3 Ø) CURRENT-MOVE
               (Y-ORIGIN 15) (X-SIZE 12) (Y-SIZE 4)
               (MOVES (VECTOR 'STONE 'PAPER 'SCISSORS)))
    (MIXINS WINDOW)
    (OPTIONS GETTABLE-VARIABLES SETTABLE-VARIABLES INITTABLE-VARIABLES))
(DEFINE-METHOD (PLAYER MAKE-MOVE)()
    (BEGIN (SET! CURRENT-MOVE  (VECTOR-REF MOVES (RANDOM 3)))
           (CAR (LIST CURRENT-MOVE))))
(DEFINE-METHOD (PLAYER TRACK-MOVE)(VAL)
    (BEGIN (COND ((EQUAL? VAL 'STONE) (SET! OP-1 (1+ OP-1)))
                 ((EQUAL? VAL 'PAPER) (SET! OP-2 (1+ OP-2)))
                 ((EQUAL? VAL 'SCISSORS) (SET! OP-3 (1+ OP-3))))))
(define-method (player show-move) ()
    (begin (CREATE) (show)
           (newline name)
           (newline name)
           (DISPLAY "   " NAME)
           (display CURRENT-MOVE name)))
;
;  CLASS DEFINITION FOR ODDS-PLAYER
;  THIS DEFINITION INHERITS FROM PLAYER
;  THE ONLY DIFFERENCE IS THE MAKE-MOVE METHOD
;
(DEFINE-CLASS ODDS-PLAYER
(instvars x y)
  (MIXINS PLAYER)
  (OPTIONS GETTABLE-VARIABLES INITTABLE-VARIABLES SETTABLE-VARIABLES))
(DEFINE-METHOD (ODDS-PLAYER MAKE-MOVE)()
    (BEGIN (set! x (max op-1 op-2 op-3))
      (cond ((=? x op-1) (set! current-move 'stone))
            ((=? x op-2) (set! current-move 'paper))
            ((=? x op-3) (set! current-move 'scissors)))
      (CAR (LIST CURRENT-MOVE))))
;
;  CLASS DEFINITION FOR AVERAGE-PLAYER
;  THIS DEFINITION INHERITS FROM PLAYER
;  THE ONLY DIFFERENCE IS THE MAKE-MOVE METHOD
;
(DEFINE-CLASS AVERAGE-PLAYER
  (INSTVARS X)
  (MIXINS PLAYER)
  (OPTIONS GETTABLE-VARIABLES INITTABLE-VARIABLES SETTABLE-VARIABLES))
(DEFINE-METHOD (AVERAGE-PLAYER MAKE-MOVE)()
    (BEGIN (set! x (MIN op-1 op-2 op-3))
      (cond ((=? x op-1) (set! current-move 'stone))
            ((=? x op-2) (set! current-move 'paper))
            ((=? x op-3) (set! current-move 'scissors)))
      (CAR (LIST CURRENT-MOVE))))
```

Listing 3.

Difference between inheritances

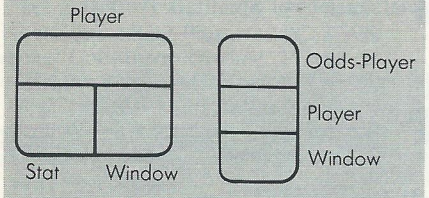


Figure 5.

exception of the Tektronix version). Inheritance allows an object to gain capabilities such as the use of windows or a way of presenting information.

Multiple inheritance allows an object to have more than one immediate predecessor. The statement *(mixins window quant)* would give the class containing this statement two approaches for presenting its data: visual and quantitative. Figure 5 shows the difference between the multilayered but still single inheritance of *ODDS-PLAYER, PLAYER, WINDOW,* and multiple inheritance.

The class definition statement of PC Scheme provides control over which variables can be manipulated. The *OPTIONS* clause of the *DEFINE-CLASS* statement gives the programmer the ability to determine which variables can be initialized, set, and retrieved. In each class definition in the listings, the *OPTIONS* clause does not include any variable names. This omission makes all variables in the object open for initialization and access through the *SET* and *GET* functions.

Changing the way we think

The implementations of the scissors-paper-stone game given in this article provide a focal point for investigating object-oriented programming. Brad Cox[5] views this style of programming as code-packaging. I believe the effect of this packaging technique extends to the basic design process and the way designers think.

Object-oriented design will soon have an impact on the development of a wide variety of systems. Elements of the class-instance way of thinking can be seen even in Ada's generics. This design technique is a natural extension of the moves to increase modularity, maintainability, and reusability of code. As systems become more complex and we seek to establish software factories, the characteristics of object-oriented design will be a basis for additional development.

Object-oriented implementations

Simula-67 introduced the earliest notions of classes and objects, but access to the content of objects was not restricted.

Smalltalk was developed by Alan Kay and Adele Goldberg at the Xerox Palo Alto Research Center, Palo Alto, Calif. The Smalltalk language operates in an interactive environment with editors, class browsers, and method compilers giving access to all parts of the system. Smalltalk is one of the most (if not the most) consistently object-oriented implementations. Even the most primitive data types, such as integers, are defined as objects.

CommonLoops is representative of several LISP-based implementations that blend object-oriented and procedure-oriented programming styles. The list-processing approach of LISP gives support for multiple-inheritance, which is not available in many of the other object-oriented languages, such as C++.

Objective-C is one of the object-oriented implementations designed as a front-end or preprocessor to a C compiler. With messaging being rated as 2 to 70 times slower than procedure calls, these products have the advantage of producing tight, fast code. Versions of these systems are tied to specific C compilers.

Knowledge Engineering Environment (KEE) is one of several knowledge programming environments that have grown out of the object-oriented arena. These represent particular applications to areas such as expert systems development. Generic capabilities for showing inheritance trees and the like are included in the system and inherited by the user's specialized objects to provide programmer support.

References

Bobrow, Daniel G., et. al. "CommonLoops Merging LISP and Object-Oriented Programming," *OOPSLA '86 Proceedings,* Association for Computing Machinery, 1986, pp. 17-29.

Cox, Brad. *Object-Oriented Programming.* Addison-Wesley, 1986.

Goldberg, Adele, and D. Robson. *Smalltalk-80: The Language and its Implementation.* Addison-Wesley, Reading, Mass., 1983.

References

1. Hayes, Brian. "Scissors, paper, stone: A tournament of Schemes," Theory & Practice, *COMPUTER LANGUAGE,* Dec. 1986, pp. 19-30.
2. Texas Instruments. *PC Scheme Language Reference Manual,* 1985.
3. Cunningham, Ward, and Kent Beck. "A Diagram for Object-Oriented Programs," *OOPSLA '86 Proceedings,* Sept. 1986, pp. 361-367.
4. Pascoe, Geoffrey A., "Elements of Object-Oriented Programming," *BYTE,* Aug. 1986, pp. 139-144.
5. Cox, Brad. *Object-Oriented Programming.* Addison-Wesley, 1986.

John McGregor is an associate professor of computer science at Murray State University, Murray, Ky. He became interested in object-oriented programming while in a summer faculty research position at Lawrence Livermore National Laboratory, Livermore, Calif.

Artwork: Anabella Gonzalez

USING THE WRONG LANGUAGE CAN BE MURDER. SPEAK SMALLTALK/V.

Let's talk languages. Programming languages like Turbo Pascal, C or Basic can be killers. To many, they're foreign, complex, and generally intimidating. Mistakes can be deadly.

With Smalltalk/V, you have an elegantly simple solution that puts the power and majesty of a major AI programming language on your PC or compatible. It makes no difference if you're an experienced programmer or just getting started. Smalltalk/V gives you an easy-to-use and flexible programming tool.

This is the same language used by leading software companies for their new product development. There are sound reasons for this. Smalltalk/V offers a totally integrated programming environment using the premier object-oriented language. You use natural language rather than complex programming codes. It puts Macintosh-type graphic features on a PC including overlapping windows, bit-mapping, pop-up menus, and a mouse interface. More than mere window dressing, Smalltalk/V delivers fully interactive windows that are easy to build and quick to modify.

But don't just take our word on it. Hear what the experts have to say:

"This is the real thing folks. A super Smalltalk like this turns your PC into a hot workstation. It's fantastic... Highly recommended."

John Dvorak
Contributing Editor
PC Magazine

"The tutorial provides the best introduction to Smalltalk available."

Dr. Andrew Bernat
AI Expert Magazine

"Smalltalk/V is the highest performance object-oriented programming system available for PCs."

Dr. Piero Scaruffi
Chief Scientist
Olivetti Artificial
Intelligence Center

Today, thousands of professionals, scientists and engineers are using Smalltalk/V to solve both simple and expert problems. Giving them a new dimension in computer applications for their PC.

Put new life into your PC by calling toll free 1-800-922-8255 and ordering Smalltalk/V today. Smalltalk/V by Digitalk, Inc., 9841 Airport Blvd., Los Angeles, CA 90045. (213) 645-1082.

$99.95

Smalltalk/V comes with 10 starter applications including Prolog and each Application Pack adds several more. All source code is included. Supports 640 x 480 color graphics with color extension pack.

Smalltalk/V requires DOS and 512K RAM on IBM PC/AT/PS or compatibles and a CGA, EGA, Toshiba T3100, Hercules, or AT&T 6300 graphic controller. A Microsoft or compatible mouse is recommended. Not copy protected.

Turbo Pascal is a trademark of Borland Internation. IBM, IBM PC/AT/PS are trademarks of International Business Machines Corporation. Macintosh is a trademark of Apple Computer, Inc.

TO ORDER CALL 1-800-922-8255 TODAY.

60-DAY MONEY-BACK GUARANTEE*

Send check, money order, or credit card information to: Digitalk, Inc., 9841 Airport Boulevard, Los Angeles, CA 90045.

Credit Card ☐ VISA ☐ Mastercard

Card number: _____

Expiration date: _____

Name: _____

Street Address: _____

City/State/Zip: _____

Smalltalk/V	$99.95
RS-232 Communications Application Pack	$49.95
EGA/VGA Color Extension Pack	$49.95
"Goodies" Application Pack	$49.95
SPECIAL OFFER: Smalltalk/V and all 3 packs only	• $199.95
Shipping and handling (Outside North America	$5.00 $15.00)
California residents add applicable sales tax	
TOTAL	$_____

*Unconditional 60-day money-back guarantee. Simply return to Digitalk, Inc. and your refund will be immediately forwarded to you.

Smalltalk/V

digitalk inc.

CIRCLE 32 ON READER SERVICE CARD

Will the Real AI Language Please Stand Up?

By Harvey Newquist III

Artificial intelligence is perhaps the most overused—and abused—buzzword in the current age of computer science. The AI arena has been divided into several factions, all of which help distort its real benefits and potentials.

On the one hand we have the popular business media, which has billed AI as everything from the greatest creation since sliced bread to a worthless endeavor perpetuated by computer hucksters.

Then we have the vendors and users, caught in a continual tug-of-war over how to properly implement AI, how to integrate it with existing applications, which machines to run it on, where it fits best in a company's computing strategy, etc.

Finally, we have the hard-core researchers—those people who gave us AI in the first place. They spend their time thinking about thinking and coming up with new ways to create machines that more closely resemble humans and human capabilities.

Though these groups are quite separate, one simple question has begun to bind them together: Which language is best for developing AI applications, particularly expert systems and natural language interfaces?

Not since the whole concept of AI was developed has there been such chaos and confusion in the marketplace over the suitability of a programming language to the needs of the developer and user. So let's look briefly at the history of languages in AI and then see what's being done with them in 1987.

How it all began

The notion of artificial intelligence was conceived in 1956, when a group of gentlemen assembled for the Dartmouth Summer Conference to discuss ways to improve the capabilities of computers.

This small group was led by Massachusetts Institute of Technology researcher John McCarthy, who presented the term "artificial intelligence" to the group as a concept that they should all work to achieve.

Not coincidentally, McCarthy was also the inventor of a brand new language called LISP (LISt Processing). McCarthy's new programming language was symbolically oriented: it shunned the standard computer language process of employing numerical processing. It was with this language that the group hoped to develop computers and computer programs that would become truly intelligent.

After this first conference, the attendees returned to their respective universities—primarily MIT, Cambridge, Mass., and Carnegie-Mellon University, Pittsburgh, Pa. Small AI labs at these two schools were fostered to work on problems in artificial intelligence, particularly expert systems and robotics.

Within a few years, however, McCarthy's LISP began to mutate. As different organizations began to use LISP, they also began to customize the language to suit their own needs.

For instance, Bolt Beranek & Newman, the research and contract company, began developing its own version of LISP, known as BBN LISP, for internal use. But some of the people working on the project left to go to the Xerox Palo Alto Research Center (PARC) in Palo Alto, Calif., and they took this LISP with them. There it evolved again and became known as InterLISP-D.

At about the same time, researchers were leaving CMU and going to Stanford University, Stanford, Calif., where McCarthy had gone after leaving MIT. More LISPs made their way around the country, and no less than two dozen versions of the language appeared, some incompatible with others.

Since this was the researchers' language of choice, it was the first one used extensively for developing AI applications. Expert systems such as the medical programs CADUCEUS and MYCIN were written in LISP, as was the pioneering molecular analysis program known as DENDRAL.

Some of these languages were developed specifically in InterLISP. Extensive use of this language was the result of a time-sharing program set up by the National Science Foundation and Stanford, where the main computer ran—you guessed it—InterLISP.

So the initial AI developers were actually a fairly tight-knit group of less than 50 researchers who all decided to use the same basic language for their work. Most of these individuals knew each other as well, so they developed somewhat of a monopoly on what was going on in AI and what languages were to be used. Thus LISP became the standard language for the AI community, with little or no competition from other forms of programming languages.

On the foreign front

Now we turn our sights to Europe. Both France and Sweden had emerging AI projects going on in the early 1970s in a number of universities, most notably Sweden's Uppsala University and France's University of Marseilles.

The primary work coming out of Europe involved logic programming, another approach to emulating human thought processes. The French research project developed a language known as PROLOG (PROgramming in LOGic). It was more structured than LISP but was better for following lines of logic.

PROLOG finally made its way across the Atlantic in the early 1980s, where it met with huge resistance from the established AI community in the U.S. Much of the time between 1981 and 1986 was spent arguing the pros and cons of one language or the other.

By 1986, LISP almost triumphed. Its proponents had kept PROLOG at bay by convincing U.S. customers that it was more of a methodology than a full-featured language. The number of companies selling PROLOG in the U.S. at that time was less than a dozen.

But PROLOG had gone to Japan. There the Japanese Fifth Generation Project—designed to create knowledge-based hardware and software—decided to use PROLOG as the basis for all of its operations. This decision was based in large part on the fact that PROLOG was not a U.S. product and the Fifth Generation was aimed directly at assaulting the U.S. claim to the highest of high technology.

AI vs. conventional languages

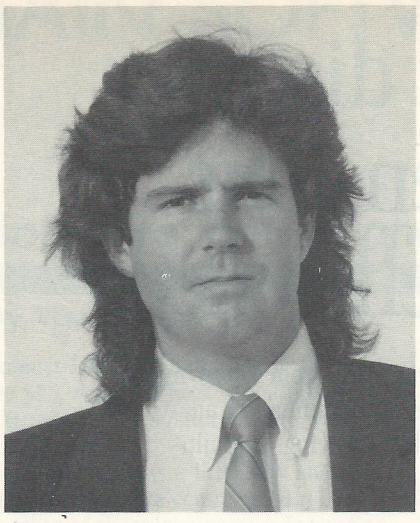
This brings us to the present state of affairs, but we have not yet looked at other languages. Just because the original AI researchers chose LISP doesn't mean that BASIC, FORTRAN, or COBOL won't work.

The difficulty with employing these languages in AI programming is that they don't have all the logic or symbolic facilities that make LISP or PROLOG a little more efficient at the development stage. But a good programmer can work around these shortcomings and create full-featured expert systems in any language.

For instance, one of the first expert system shells for the PC was developed by the U.K.'s Donald Michie in FORTRAN. Upon its introduction, this package—known as Expert-Ease—was the most successful PC tool available.

But languages of all sizes and shapes can be used for AI, and not just at the PC level. A division of Cullinet, Distributed Management Systems, has a COBOL expert system for IBM environments known as IMPACT. And IBM itself has an internal expert system for teaching music theory that was written in Pascal. With a little imagination, any language can be appropriate, given the programmer's familiarity with all its features and idiosyncracies.

Another way expert systems and various AI applications can be developed is through the use of object-oriented languages such as Smalltalk, Flavors, and OPS. Each of these has been employed for the creation of some very comprehensive applications, such as XCON, the infamous configuration expert used at



Digital Equipment Corp.

XCON was developed in 1980 by DEC and CMU and was written in OPS5, an object language developed at CMU. XCON is believed to be the largest expert system running in day-to-day use anywhere in the world.

Many organizations see object orientation as a way to add intelligent capabilities to existing languages and create an enhanced programming environment. Object-oriented languages do not require the complete relearning of a whole separate language from the one that the programmer is already fluent in. In just the last two years, object orientation has grown from an AI curiosity into a very important part of programming at large.

Winds of change

So here we are in 1987. A major shift in control of the AI community has taken place, from a small group of researchers to the users.

Along with this shift has come a marked change in what types of languages are being used. There really aren't that many fully trained LISP hackers out there, simply because there hasn't been the time to train them. Plus, the specialized machines used to run LISP are expensive and require good, experienced programmers.

Users began looking for alternatives to LISP in early 1986 and saw that with a little work they could use C for their applications—or even run LISP on standard, general-purpose machines. This gave users a choice that enabled them to save money and use their resources more economically.

As a result, the LISP vs. PROLOG argument has become moot; now it's C vs. LISP. PROLOG appears to be taking a back seat to this new debate, but not for lack of trying. Case in point: Borland International's Turbo PROLOG has allegedly sold close to 100,000 copies since its introduction last year, allowing a whole new generation of programmers to become acquainted with the language.

But the pervasiveness of C and its relationship to UNIX, coupled with the voices of LISP vendors and researchers, has made these two languages the center of AI attention.

Some expert system companies have dropped LISP altogether, focusing on more flexible languages. Teknowledge, for instance, switched its LISP-based S.1 tool to C in late 1985 and now only supports its C versions. Inference Corp. and The Carnegie Group are still keeping the LISP versions of their expert system tools but plan on having C implementations available by the end of this year. Plus, the recent demise of LISP Machine Inc. and the troubles at LISP leader Symbolics have not made things any easier for those organizations trying to decide on which path to follow in developing AI applications.

On the other hand, a number of companies such as Lucid, Sun Microsystems, Apollo, and Franz Inc. have all attempted to make it easier for the mass market to use generally available machines and tools for AI development instead of relying on specialized machines and tools. These companies are focusing on general-purpose environments such as UNIX to help the spread of AI development.

So now we have C vs. LISP. Which is better? Depends on what you need. LISP is a tough language to port to a lot of machines; that happens to be one of C's strengths. C isn't so great at some of the screen interfaces found in a majority of expert systems; LISP has enough interface capabilities to keep even the most jaded programmer happy. And the list goes on.

The fact remains, however, that AI is not any particular language nor any particular programming style. No AI language is required for developing applications, just as no specific brand of gasoline is required to get your car running.

Developing an expert system or natural language application all boils down to a few very simple things. First, how comfortable is the programmer with a specific language? Second, what kind of machine is the application going to run on? Third, how much is the use of a certain language going to cost? (Management may argue that this should be the first consideration.)

And last, but certainly not least: Does it work?

Harvey Newquist III's column In Practice appears each month in AI EXPERT. *In addition, he is editor of the* AI TRENDS *newsletter and a contributing editor to* COMPUTERWORLD.

Let us help you discover the many dimensions of artificial intelligence.

AI expands our ability to work creatively with computers. From improving manufacturing methods to exploring the mysteries of the human mind, AI constantly offers new possibilities for creative problem solving. And ultimately it may change our perceptions.

We publish the leaders.

Addison-Wesley's list of AI authors and contributors reads like a who's who in artificial intelligence. Patrick Henry Winston, Donald Waterman, Adrian Walker, Paul R. Cohen, Edward A. Feigenbaum, Avron Barr, Edward Shortliffe, Eugene Charniak. . . and many others who've made important contributions to the field. For over a decade, we've made it our business to bring you books by the most innovative minds in AI.

You know you can depend on Addison-Wesley to supply you with the very best books on the history and development of AI, the latest research developments, languages, and expert system tools and techniques. And we can show you how expert systems have been and will be put to use in areas as diverse as financial planning, telecommunications, electronics, oil exploration, statistics, and many more.

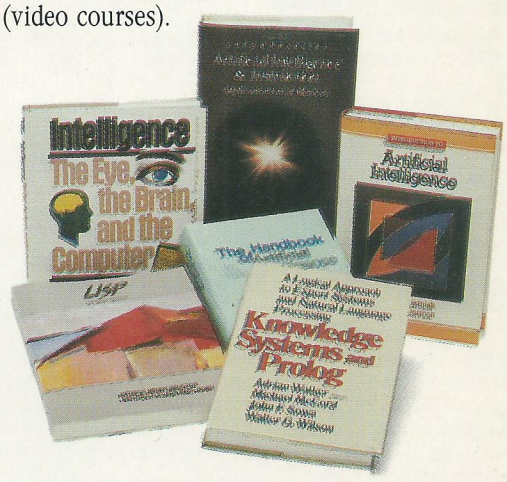
And video courses by the experts.

We've joined forces with respected AI leaders Patrick Henry Winston, Randall Davis, J. Ross Quinlan, and Donald Waterman to produce three video-based courses on the practical applications of AI technology.

These courses introduce AI methods and concepts into your organization, help you plan and implement a successful knowledge-based system, and develop and refine your knowledge acquisition techniques. In addition, we can recommend qualified facilitators to conduct the courses on a consultation basis.

Promoting excellence in AI education . . . in any medium.

Addison-Wesley is much more than a book publisher. Think of us as a knowledge systems company too. To receive more information or a free catalog of our AI books and video-based courses, please write to Dept. P-006. Or call and ask for Denise Descoteaux, ext. 2630 (books), or Training Systems, ext. 2769 (video courses).



Your number one source to the finest minds in artificial intelligence.

Addison-Wesley

Reading, MA 01867
(617) 944-3700

CIRCLE 31 ON READER SERVICE CARD

POWERLISP

THE PROFESSIONAL AI ENVIRONMENT

Run Larger Applications On Smaller Machines.

PowerLisp™ supports complex AI applications today. PowerLisp™ is the first complete Professional AI development environment for PC's*. With PowerLisp,™ it is now possible to develop and run serious professional 60 megabyte applications on 3 megabyte PC machines!

Achieve The Speed, Power And Performance Of A Symbolic Processor On Any 3 Megabyte PC AT Or 386.

PowerLisp™ is the only PC-based AI Environment to offer virtual memory. By reallocating usable address space via 286/386 Address Translation Hardware, virtual memory allows development and implementation of full 60 megabyte applications on a PC/AT compatible, or Intel™ 386 processor, with all of the speed, power and performance characteristics of a mainframe or specialized AI machine.

No longer will lack of address space (or expense of hardware) limit your application! The additional program space virtual memory provides enables the creation of applications never before thought possible for a PC-based system.

Virtual Memory Is The Answer For AI On A PC.

By nature, the better AI applications tend to be large, and before PowerLisp,™ had to run on expensive dedicated systems, such as mainframes or symbolic processors. The decrease in speed and performance normally associated with running large AI applications on PC based systems has been eliminated with PowerLisp.™

Your Development System Is Your Delivery System.

The price/performance ratio PowerLisp™ delivers makes it a very attractive alternative to many other more costly AI environments, especially in cases where the application you develop needs to be delivered to users at low cost.

The Complete Implementation Of InterLisp™ For The PC AT Or 386... And More.

InterLisp,™ originally developed for use on a PDP-10, has now been successfully ported to run on the microcomputer. With virtual memory, PowerLisp™ is, without question, the most powerful and most complete professional development environment for the PC. PowerLisp™ also contains many of the features of Common Lisp such as packages, multiple-value returns, catch and throw forms, etc.

A Complete, Professional Development Environment.

Even though development is performed on a PC, PowerLisp™ offers all of the high-level tools necessary to develop professional level AI applications quickly and efficiently. Aside from the conventional array of sophisticated program editing and debugging features, PowerLisp™ includes high-level tools such as DWIM (or Do What I Mean), which automatically corrects many typos and entry errors; Masterscope, a unique static program analyzer; CLISP a complex and useful iteration facility; and Programmer's Assistant, which allows commands to be redone or undone. With all of it's advanced tools, PowerLisp™ is actually easier to use than many conventional programming languages.

Large AI Applications Have Already Been Ported To PowerLisp.™

PowerLisp™ has been chosen as the implementation language for multiple applications.

Two particularly complex applications include THEMIS™ (a natural language query system), and EMYCIN, an expert system shell. THEMIS™ provides the user with the ability to query databases in English from a workstation. It incorporates the extensive windowing features of PowerLisp™ and utilizes the virtual memory capability. EMYCIN is an implementation of the expert systems shell originally developed at Stanford University. EMYCIN, extended by the addition of a windowed interface, takes advantage of the high-speed throughput and virtual memory available in PowerLisp™ to process large rule bases.

PowerLisp™ Supports Both The 286 And The 386.

If you find it difficult to choose between the 286 today, or wait for the 386 hardware to become available tomorrow, keep in mind that the 286 version of PowerLisp™ is fully compatible with the 386 version. The 286 application you develop today will not become obsolete for the 386. The 386 version directly supports the 386 instruction set and memory management hardware today. Very fast on a 286, PowerLisp™ will run up to 6 times faster on the 386!

Interested, But Need More Memory?

A PowerLisp™ option is available bundled with a 3 megabyte memory expansion board priced at $1695, or $1195 without the expansion board. The 386 version is priced at $1695.

Put PowerLisp™ To Work For You.

PowerLisp™ is the complete professional development environment you need to create complex AI applications on a PC. With virtual memory, PowerLisp™ is the ideal application to maximize price/performance ratios in development as well as delivery. Call our toll-free number today to order and for more information.

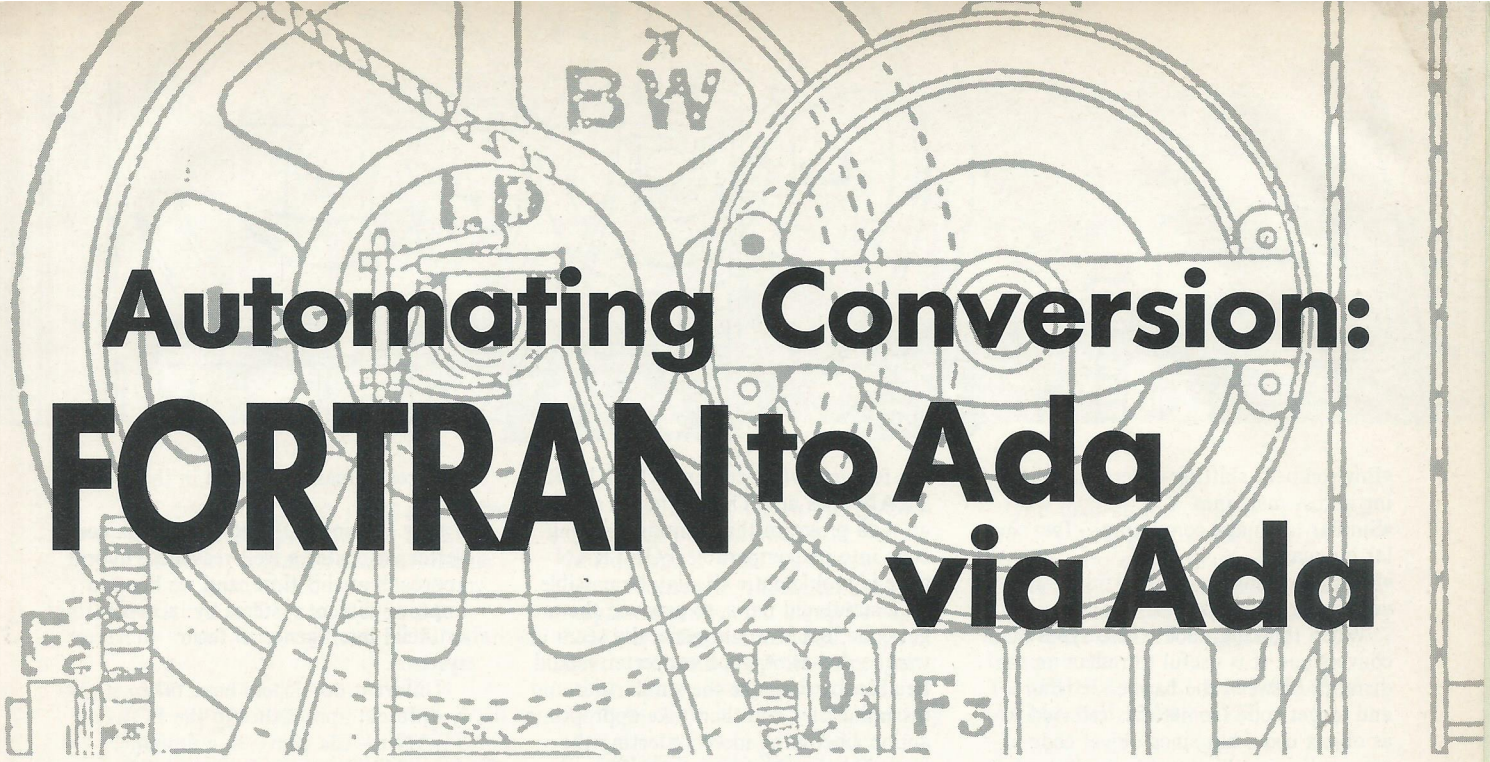
Major credit cards are accepted.

See Parrot (a custom training/sales demo application) and PowerLisp® at booth #111, AAAI Conference/Seattle.

*PowerLisp™ requires a PC AT or 386 with 1.5 megabytes of internal memory and a 20 megabyte hard drive.

CIRCLE 30 ON READER SERVICE CARD

MicroProducts
INCORPORATED
370 W. Camino Gardens Boulevard
Boca Raton, Florida 33432
(305) 392-9800
1-800-553-0777



Automating Conversion: FORTRAN to Ada via Ada

By Lance B. Eliot and Floyd L. Holliday

The development and maintenance of software is hampered by a lack of robust tools for the software professional. One research effort to create such tools is the Knowledge-Based Systems Factory (KBSF) project; this article describes one tool that fits into the systems factory concept. The tool described is used to convert FORTRAN programs into Ada programs and is itself written in Ada.

Though the KBSF approach to the conversion process is not unique, many facets of the converter provide interesting insights into the similarities and differences of FORTRAN and Ada. As Ada becomes more widely adopted there is bound to be increased attention from traditional FORTRAN programmers as to how to develop Ada programs.

We hope to illuminate some of the dangers in creating blind Ada code based on FORTRAN reasoning, in the same way that the KBSF converter produces comparable Ada code for a given segment of FORTRAN code (functional but not elegant). Finally, we provide some speculation on the need to incorporate intelligence into a language-to-language converter such as FORTRAN-to-Ada, especially if full conversion is desired (such as producing true targeted code that is both understandable and maintainable).

The right tools

Much of the literature on software engineering decries the current set of tools available to software professionals.[1,2] Software can be a simple artifact or it can be a massive, tangled web of delicately balanced lines of code waiting for just the worst moment to become unreliable, frustrating the end user and developer alike.

What software professionals want is the same thing many other professions already have: tools to help them get their job done. But these tools are not of the manual variety; instead, the software professional desires automation to help produce automation.

Recent attempts to develop computer-aided software engineering (CASE) have met with moderate success.[3-5] The KBSF research project has explored the use of a production factory metaphor in an effort to develop CASE.[5] The Japanese have also adopted the factory notion for software development, but their use of the factory metaphor has been criticized as stifling creativity.[6]

In the KBSF a software professional is portrayed as a skilled craftsperson while the software factory deals with knowledge-based tools that "can focus on sophisticated machinery, manufacturing processes, and strategies for effectively organizing their use by large development teams."[6]

Automation in the KBSF assists, rather than replaces, skilled software engineers. Furthermore, the tools are of the heavy machinery type and are intended for large-scale software development.

Some of the tools in the current factory toolkit include:

- Language-directed editor generator
- Flow analyzer and testing system
- Configuration management system
- Window-based user interface and command processor
- Gist specification analyzer and simulator
- Document integration facility
- Spreadsheet application generator
- Project planning and scheduling system
- Knowledge-based project management support system

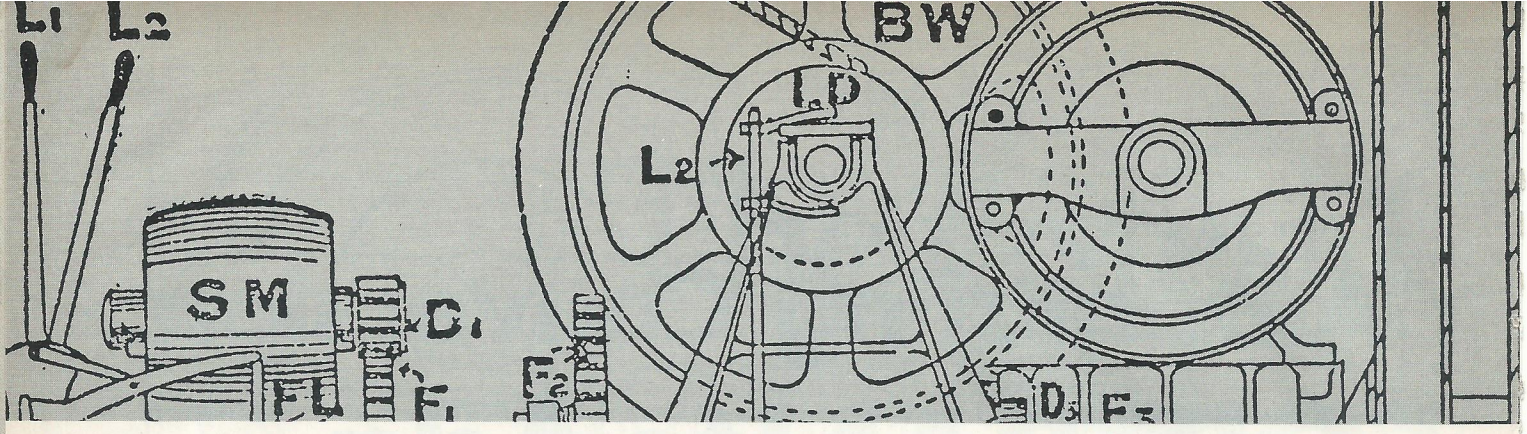
The KBSF is an incremental development effort, spanning several years of evolution. Since the project is part of a university research effort, the KBSF has been largely developed by graduate students under the direction of faculty.

Conversion and distance

One piece of work related to the KBSF, conversion, is the subject of this article. This word and process often strike fear in the hearts of many a software developer. Language-to-language converters have been around for years, and this section details some of their history. Following this discussion is a description of our FORTRAN-to-Ada converter. Finally, we consider the future directions for language-to-language converters.

The process of conversion can differ significantly depending upon the distance between the base and target languages. A range of distances is conveyed by considering some of the types of conversions possible:

- Formatting—Minor textual conversion
- Version changing—Converting within a language



•Environment shifting—Version changing across machines
•Similar language converting—Two similar languages
•Dissimilar language converting—Two very different languages

When thinking about these types of conversions, it is useful to reflect on the distance between the base code (source) and target code (sometimes referred to as object code, but since object code often implies machine code, we choose not to use the term here). Typically, a tool that formats a program doesn't require any great sophistication, and the base and target sets of code are different in only minor, textual fashion (indentation, blank lines, longer variable names, pagination, added notations, etc.).

The intermediate distances, such as version changing and environment shifting, are conversions between a base in language X and a target, also in language X. Thus we might have to convert FORTRAN II code to FORTRAN IV code or VAX FORTRAN to IBM FORTRAN. In some cases the distance between these same-language entities is quite small (approaching a formatting problem), while in other cases the distance is quite large.

Finally, conversion between two different languages can vary depending upon how similar or dissimilar the two languages appear to be. Some languages, such as a typical BASIC or FORTRAN, are fairly similar. In the latter case, much of the conversion process can rely on simple syntax changes. Yet even with a typical BASIC or FORTRAN the conversion can quickly become complicated if programmer tricks are allowed in the language.

For example, a BASIC program might be written in such a fashion as to take advantage of certain machine dependencies (16-bit words, I/O address locations, etc.). Trying to convert a tricky program into its comparable target code can be difficult—both for a human and a machine.

Other conversion issues

Some conversions are not practical at all. For example, Ada has an inherent tasking feature, while current ANSI FORTRAN does not. Thus trying to convert an Ada program that contains tasking code into its comparable FORTRAN code is problematic (it may be possible, on a simulated basis, to produce the target code, but it would generally seem unwise to do so). A good converter should be able to recognize the similarities and dissimilarities and then take appropriate action (including merely alerting the user that the change may not be practical).

Some conversions produce comparable code but do not exploit the capabilities of the target language. Again, consider a straight, sequential FORTRAN program that is being converted into Ada. Normally the resulting code would not exploit the tasking feature of Ada. Thus, providing strict conversion may not be desired. Instead, the converter might be required to redo the base so that it is improved in the target language.

Unfortunately, an awkward trade-off begins to emerge as the absolute difference between the produced base and target begin to stretch sensible limits. If a programmer tries to compare the base and target, the two may seem to be entirely different programs! At the point where the base and target vary to such a large degree, a person using the conversion may find that previous internally focused documentation has become obsolete—making maintenance a tougher job than it may have been to begin with.

Many of these topics generate a host of concerns. Conversion is an interesting problem related to compiler theory and other computer science topics. For a discussion of additional issues in the conversion debate, see references 2 and 7. A list of several commercial conversion products is also given in reference 7.

Converting FORTRAN to Ada

Ada is the new kid on the block, FORTRAN the old. Organizations wishing to adopt Ada often will already have an entrenched investment in existing FORTRAN code.

One approach to what to do with all that FORTRAN code is to convert it into Ada code, either by hand or machine. Currently, even the best FORTRAN-to-Ada automatic converter still requires human assistance in the tougher cases.

Our current FORTRAN-to-Ada converter advocates a well-reasoned, hybrid approach at this time, and we hope to improve current systems by incorporating artificial intelligence in future converter systems.

Different developers have taken slightly different approaches in the FORTRAN-to-Ada conversion design.[8-11] Arguably, the most significant difference revolves around the issue of how close the target should logically be to its original base. J. Slape and P. Wallis[10-11] tend to favor not maintaining the new Ada, as though it was still in its base FORTRAN version (in other words, the converter can produce target code that is dramatically different from its given base).

Our philosophy is to tread a thin line between producing the closest equivalent code and the its-not-the-same semantically farther away code. We want to produce generated Ada code that is understandable and maintainable in the sense of the original FORTRAN code.

Our wish list continues. We do not want to undermine the portability and efficiency of the FORTRAN code. We do not want to produce Ada code that is much bigger than the base FORTRAN. The design of the FORTRAN program should be reusable with the new Ada code. These goals, and others, present a formidable barrier.

The KBSF approach

Choosing which language to write the converter in is often a problem itself. The KBSF converter is written in Ada. We wanted a tool that would incorporate many of the preferred aspects of good software, and Ada met our criteria (portability, understandability, and maintainability). As a result, the converter also serves an obviously useful purpose if it is selected by adopters of Ada.

The Ada we used is the ANSI standard described in reference 12; when we refer to FORTRAN we mean the ANSI

HOT GRAPHICS PACKAGE FOR C PROGRAMS.* $39.95

Everything you need to write dramatic graphics effects into your Eco-C88 C programs. Some of the features include:

- Support for EGA, CGA, and Z100
- Over 100 graphics and support functions, many of which are PLOT-10 compatible.
- Many low level support routines reside outside your small model code-data area
- Can write dots thru the BIOS (for compatibility) or to memory (for speed)
- Graphics function help from CED editor available
- World, pixel or turtle color graphics modes
- 47 standard fill patterns, 17 line dashing patterns, Hershey fonts, plus user defineable fill, dash and fonts
- Supports view areas, rotateable fonts, clipping, arbitrary fill areas, extensive error checking, examples, and user's manual.

A must for the graphics enthusiast and a bargain at only $39.95

***Requires Eco-C88 C Compiler.**

NEW POP-UP WINDOWS FOR YOUR C PROGRAMS.

This windowing library allows you to add pop up windows in your C programs quickly and easily. Use them for help windows, selection menus, error messages, special effects—anywhere you need an attention getter. Just some of the features include:

- CGA, EGA, and monochrome support
- Slow mode option for "flicker" displays
- Control any program that goes through the BIOS
- Use up to 255 windows
- No special window commands; use print f ()
- Resize and move windows
- Custom window titles and borders
- Can be used with ANSI device driver
- Most of window's code-data lies outside small model limits
- Use any of the IBM text or block characters
- User's manual and examples

The Windowing Library requires an IBM PC compatible BIOS and the Eco-C88 C compiler.

ONLY $29.95

HANDY LIBRARIAN MAKES LIFE EASIER.

Now you can combine your modules, functions, and subroutines into your own library for easy link commands. Fully compatible with ANY standard OBJ format files (not just Ecosoft's products). With the Ecosoft librarian, you can:

- Add, delete, and extract from a library
- Get table of contents or index of a library
- Combine libraries, control library page size, use switches for combinations, process complex library requests, use wildcards, and do library directives from command files.
- Complete with user's manual

A valuable addition for any programmer.

ONLY $29.95

THE FIRST PROFESSIONAL 'C' COMPILER FOR UNDER $60.

A C compiler with many ANSI enhancements at an unbelievably low price. The Eco-C88 C compiler has:

- Prototyping (the new type-checking enhancement)
- Enum and void data types
- Structure passing and assignment
- All operators and data types
- A standard library with more than 200 functions (many of which are System V compatible for greater code portability)
- CC and mini-make that all but automates the compile process
- 8087 support (we sense the 8087 at runtime — no dual libraries)
- ASM or OBJ output for use with MSDOS linker
- Tiered error messages — enable-disable lint-like error checking
- Fast compiles and executing code
- Expanded user's manual
- CED full-screen program editor

Everything you need at the unbelievable price of $59.95.

Eco-C88 C compiler requires an IBM PC, XT, or AT (or compatible) with 256K of memory, 2 disk drives and MSDOS 2.1 or later.

Orders only:
1-800-952-0472
Technical Information:
(317) 255-6476

NOT COPY PROTECTED.

Ecosoft Inc.
6413 N. College Ave.
Indianapolis, IN 46220

ORDER FORM CLIP & MAIL TO: Ecosoft Inc., 6413 N. College Ave., Indianapolis, IN 46220

ITEM	PRICE	QTY	TOTAL
Flexi-Graph Graphics	$39.95		
Window Library	$29.95		
Eco-Lib Librarian	$29.95		
Eco-C88 C Compiler CED	$59.95		
		SHIPPING	
		TOTAL (IND. RES. ADD 5% TAX)	

PAYMENT: ☐ VISA ☐ MC ☐ AE ☐ CHECK

CARD # _____ EXPIR DATE _____

NAME _____

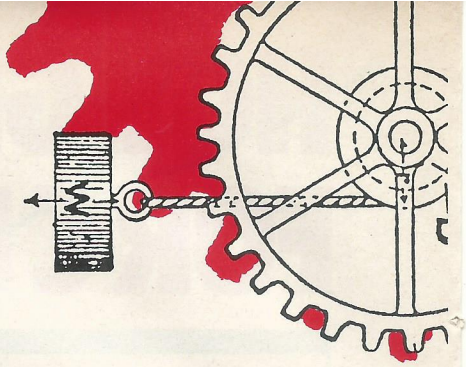
ADDRESS _____

CITY _____ STATE _____

ZIP _____ PHONE _____

ECOSOFT

CIRCLE 33 ON READER SERVICE CARD



```
Sample FORTRAN base containing functions

        program testl
        integer funtst
        integer array(3,5)
        array(1,2) = array(3,4)
        i= funtst(3,1)
        stop
        end
        integer function funtst(a,b)
        integer a,b
        if (a .gt. b .and. a+l .gt. b .or. a-l .le. b) then
          funtst = a + (b-b)
        elseif (a .gt. b .and. a+l .gt. b .and. a+2 .gt. b) then
          funtst = a + -b
        endif
        if (a .gt. b .and. a+l .gt. b .or. a-l .le. b) goto 3
          funtst = a+(b*b)
          goto 4
3       continue
          funtst = (a+b) * b
4       return
        end
```

Listing 1A.

```
Sample Ada conversion generated from 1A

-- The main program is:
with FORTRAN; use FORTRAN;
        procedure TEST1;

-- The procedure and functions are:
with FORTRAN; use FORTRAN;
        function FUNTST(A : in INTEGER;
                        B : in INTEGER) return INTEGER;

with TEXT_IO;
with FUNTST;
procedure TEST1 is -- main program
  N_ARRAY : array(1..3,1..5) of INTEGER;
  I : INTEGER;
begin
  N_ARRAY(1,2) := N_ARRAY(3,4);
  I := FUNTST(3,1);
  raise FORTRAN_STOP;
exception
when FORTRAN_STOP =>
  TEXT_IO.PUT_LINE("FORTRAN STOP");
end TEST1;
with TEXT_IO;

function FUNTST(A : in INTEGER;
                B : in INTEGER) return INTEGER is
  FUNTST_RESULT : INTEGER;
begin
  if (A>B and A+1>B) or A-1<=B then
    FUNTST_RESULT := A+(B-B);
  elsif A>B and A+1>B and A+2>B then
    FUNTST_RESULT := A+(-B);
  end if;
  if (A<=B or A+1<=B) and A-1>B then
    FUNTST_RESULT := A+(B*B);
  else
    null;
    FUNTST_RESULT := (A+B)*B;
  end if;
  return FUNTST_RESULT;
end FUNTST;
```

Listing 1B.

X3.9 1978 standard.[13] Other converters sometimes pick specific vendor versions of Ada or FORTRAN, but we chose the standards.

FORTRAN tends to describe its modular structures in one set of terms, Ada in another. Thus we use the word "subprograms" to refer to Ada subprograms and FORTRAN procedures, "procedures" to refer to Ada procedures and FORTRAN subroutines, and "function" to refer to the same module types in both languages.

FORTRAN has essentially two fundamental data types: scalar and array. The FORTRAN scalar class is considered an atomic element in our view. Ada has both predefined types and program-defined types. The FORTRAN type *COMPLEX* (an example of an atomic element in FORTRAN) is viewed as a compound, program-defined record type in our targeted Ada code.

How the converter works

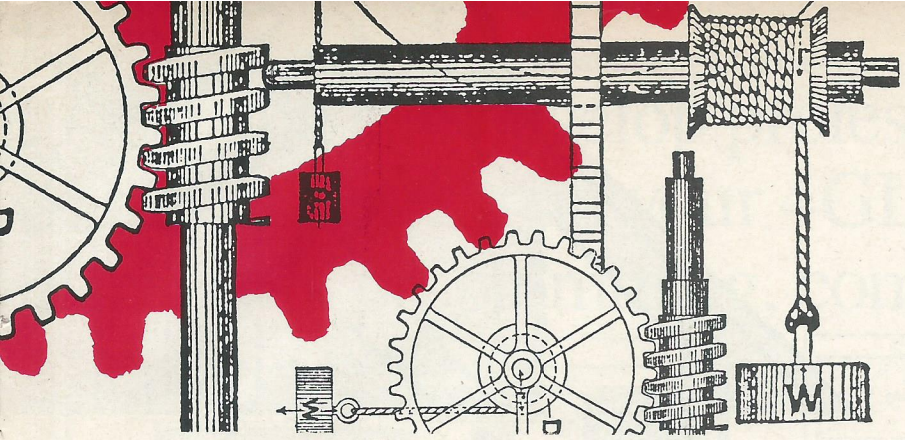
The KBSF converter makes two passes through the FORTRAN base. Unlike some other converters, no prior modification of the FORTRAN code is necessary—we assume the FORTRAN code is in its pure form (possibly a mess).

A FORTRAN compiler might be expected to compile subprograms independently, but in our case this is not a sensible approach. The entire FORTRAN code, including the main program and all of its associated subprograms, is used during the conversion process.

The FORTRAN code is actually read twice, once during each pass. In the first pass the code is scanned and an intermediate form produced. As part of the intermediate form, a special deep structure that reflects various aspects of the FORTRAN base is created. The deep structure is then analyzed for global consequences.

During the second pass the deep structure is enhanced (details added in) and the Ada target produced. This is accomplished by walking in a preordered manner through the deep structure.

Parsing is done via recursive descent. FORTRAN can be a tough language to



parse due to its lack of reserved words and the difficulty of identifying tokens. (Remember the old story about the *DO* loop that allegedly led to a missile mishap, wherein the *DO* loop was supposed to be *DO 100 I=1,20* and was entered as *DO 100 I=1.20*. The latter is an assignment statement, not a *DO* loop.)

The KBSF parser constructs a tree using the FORTRAN syntax and some semantic additions. Consider a parser encountering a FORTRAN name and parenthesized list. Is this an array variable, function call, or possibly a procedure call? We place the name into the tree as a *vrbl* and then use later context and other information to determine its actual purpose.

An Ada package called FORTRAN is used to add Ada features normally found in FORTRAN, including *SQRT* and *COMPLEX*.

Ada generally discourages the use of global *Common* (for a series of good reasons), but many FORTRAN programs make extensive use of *Common*. To adhere to our philosophy regarding FORTRAN-to-Ada conversion, we use Ada packages to handle the *Common* feature.

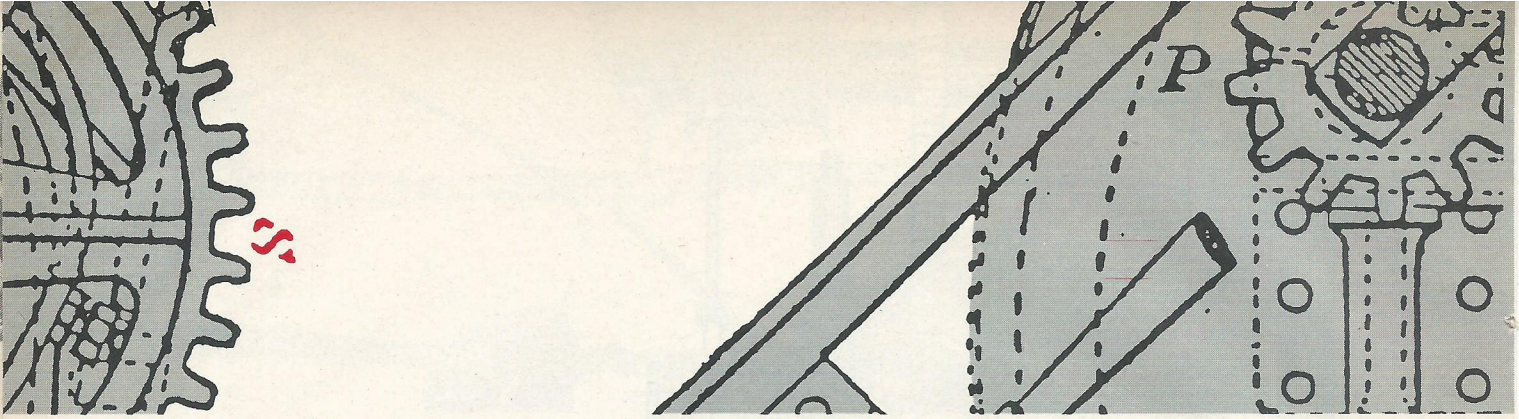
FORTRAN subroutines become li-

```
Sample FORTRAN base illustrating GOTOs

            program gotos
            integer x, y
            if (x .ne. 1) goto 100
              y=1
            goto 500
100         if (x .ne. 2) goto 200
              y=2
            goto 500
200         if (x .ne. 3) goto 300
            if (y .ne. 0 .and. y .ne. 3) goto 250
            z=0
            goto 270
250         z=y
270         y=3
            goto 500
300         if (x .ne. 4) goto 400
              y=4
            goto 500
400         y=x
500         continue
            if (x .ne. 5) goto 600
            y=5
600         stop
            end
            subroutine more
            if (x .eq. 0) goto 20
              y=y+1
              goto 30
20            y=y-1
              if (y .eq. 0) goto 25
                z=1.
                return
25              z=-1.
                return
30          return
            end
```

Listing 2A.

```
Sample Ada conversion generated from 2A

  -- The main program is:
  with FORTRAN; use FORTRAN;
           procedure GOTOS;

  -- The procedures and functions are:
  with FORTRAN; use FORTRAN;
           procedure MORE;

  with TEXT_IO;
  procedure GOTOS is -- main program
    X : INTEGER;
    Y : INTEGER;
    Z : FLOAT;
  begin
    if X=1 then
      Y := 1;
    else
      if X=2 then
        Y := 2;
      else
        if X=3 then
          if Y=0 or Y=3 then
            Z := FLOAT(0);
          else
            Z := FLOAT(Y);
          end if;
          Y := 3;
        else
          if X=4 then
           Y := 4;
          else
           Y := X;
          end if;
        end if;
      end if;
    end if;
    null;
    if X=5 then
      Y := 5;
    end if;
    raise FORTRAN_STOP;
  exception
  when FORTRAN_STOP =>
    TEXT_IO.PUT_LINE("FORTRAN STOP");
  end GOTOS;
  with TEXT_IO;
  procedure MORE is
    X : FLOAT;
    Y : FLOAT;
    Z : FLOAT;
  begin
    if X/=0 then
      Y := Y+1;
      else
      Y := Y-1;
      if Y/=0 then
        Z := 1.0;
        return;
      else
        Z := -1.0;
        return;
      end if;
    end if;
    return;
  end MORE;
```

Listing 2B.



brary-level Ada procedures; FORTRAN functions become library-level Ada functions. In certain cases an additional package is required, and *Use* in Ada attaches the rest of the converted function or subroutine.

Both for clarity and necessity, the Ada target contains a handy set of specifications preceding the body of the subprograms. Furthermore, a *With* clause is needed at the beginning of each subprogram body. Again, it is necessary for Ada and also helpful since it identifies subprograms actually referenced.

Additional details

Brevity here prevents our describing all the converter's details. Basically, the converter is concerned with *Common* and *Equivalence,* parameter modes, arrays, allocating variables, control stuctures (including labels, *GOTO, IF,* etc.), expressions, error handling, input and output, files, and a myriad of other details.

Let's examine three sample conversions. Listing 1A shows a FORTRAN program that illustrates a simple use of functions. Listing 1B shows the target Ada code produced by the KBSF converter and illustrates the handling of types, functions, and simple control structures.

Listing 2A shows a FORTRAN base that uses *GOTO*s, and Listing 2B contains the new Ada conversion. Notice the changes made to preserve the FORTRAN control paths but also produce a structured Ada eqivalent.

Listing 3A shows a FORTRAN pro-

```
Sample FORTRAN base illustrating
strings

  program chars
  integer names(5)
  text = 'abcd'
  names(3) = 'ef'
  text = names(2)
  i = 12
  end
```

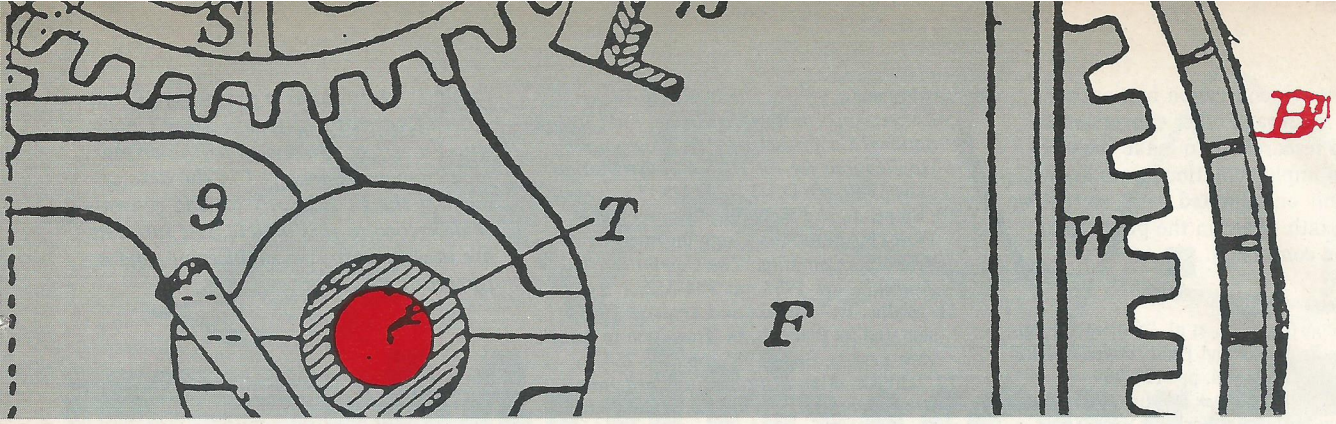
Listing 3A.

```
Sample Ada conversions generated from 3A

  -- The main program is:
  with FORTRAN; use FORTRAN;
          procedure CHARS;

  -- The procedures and functions are:

  with TEXT_IO;
  procedure CHARS is -- main program
    NAMES : array(1..5) of STRING(1..2);
    TEXT : STRING(1..4);
    I : INTEGER;
  begin
    TEXT := "abcd";
    NAMES(3) := "ef";
    TEXT := NAMES(2);
    I := 12;
  exception
  when FORTRAN_STOP =>
    TEXT_IO.PUT_LINE("FORTRAN STOP");
  end CHARS;
```

Listing 3B.

```
Data structure used in the converter

type statement_node (statement_type: statement_types) is
  record
    next_statement: statements:=null;
    label: integer:=0;
    cut_off: boolean:=false;
    comment: comments;
    case  statement_type  is
      when  assignment_statement =>
        lhs:  vrbls:=null;
        rhs:  expressions:=null;
      when  goto_statement =>
        destination: integer;
      when  stop_statement =>
        simple_stop: boolean:=true;
        stop_expression: expressions;
      when  logical_if_statement =>
        condition: expressions;
        consequence: statements:=null;
      when  do_statement =>
        do_end: integer;
        do_index: vrbls;
        m1,m2,m3: expressions;
        do_action: statements:=null;
      when  block_if_statement =>
        block_condition: expressions;
        block_consequence: statements:=null;
        block_else: statements:=null;
        block_elseif: statements:=null;
        block_last_elseif: statements:=null;
        block_falls_through: boolean:=true;
      when  block_elseif_statement =>
        block_elseif_condition: expressions;
        block_elseif_consequence: statements:=null;
      when  call_statement =>
        subprogram: vrbls;
      when  unrecognized_statement =>
        source_length: integer range 0..config.max_source_length;
        source: string(1..config.max_source_length);
      when others =>
        null;
  end case;
end record;
```

Listing 4.



gram involving strings, and Listing 3B shows the Ada eqivalent. Note in the new Ada code that the converter has figured out certain aspects of string usage in the FORTRAN code and produced very matching Ada code.

Listing 4 contains one of the data structures used in the Ada code (part of the deep structure). Listing 5 provides a segment of the converter that scans the FORTRAN source during the first pass. Listing 6 shows part of a segment that emits certain Ada code during the second pass. Listing 7 shows an internal debugging feature built into the converter (to be used by maintainers of the converter).

In principle, the converter can handle most FORTRAN. The design covers all strict FORTRAN, but the current implementation handles typical cases rather than all of the special cases. We have tested the converter with a special set of test cases and are proceeding to test it with live, full-blown FORTRAN programs.

As already briefly mentioned, in a number of subtle instances the converter will confront problematic situations. How is the converter to handle these situations? Our current approach is to have the converter issue warnings and tips regarding the suspected code.

A future approach would be to produce the correct and best possible code. We are beginning to experiment with expert systems technology added into the converter to bring about this goal.

Where it can go from here

We have examined how Ada code is developed, how FORTRAN is converted to Ada, and how code produced by a converter is modified. We hope to distill the rules and inferences exhibited in these explorations into a special converter expert system. Using our current converter, this expert system would work *with* the converter as opposed to replacing it (allowing the converter to do the grunt work and the expert system to take the finer cases).

We have discovered that the adage "FORTRAN programmers will produce FORTRAN-looking Ada code" is often quite true. Just as our converter tends to stick to the FORTRAN base, so do humans rely on their mental base of FORTRAN programs.

Of course there is much more danger in people exhibiting such behavior since they will be developing new code and must perform at levels far beyond the mere act of converting existing old code. We have found that exposing FORTRAN programmers to a converter helps make them more aware of their own biases. And, as might be expected, a programmer perceives the converter to be a piece of automation that can easily be challenged by human skills, thus motivating some FORTRAN programmers to become more Ada-like in producing Ada code.

We urge interested readers to consider how artificial intelligence may help

```
Converter segment from first pass

function parse_statement return boolean is
begin
  debug.note("parse_statement");
  if its_an_assignment and not commas then
    return parse_assignment;
  elsif cursor > length then
    return parse_blank;
  elsif parse("GOTO") then
    return parse_goto;
  elsif parse("STOP") then
    return parse_stop;
  elsif parse("IF(") then
    return parse_if;
    :
    :
    :
  elsif parse_type_name then
    if expecting_statement = title_statement  and then
     parse("FUNCTION")  then
       return parse_function;
    else
       return parse_type_statement;
    end if;
  elsif expecting_statement = title_statement and then
    parse("FUNCTION") then
      result_type:= TBD_type;
      result_star:= Ø;
      return parse_function;
  elsif parse("END") then
    syntax_requires(cursor>length);
    raise end_of_subprogram;
  else
    result_statement:= new statement_node
     (statement_type => unrecognized_statement);
     result_statement.label:= label;
     result_statement.source_length:= length;
     result_statement.source:=image;
    return false;
  end if
end parse_statement;
```

Listing 5.

break through conversion bottlenecks. We also encourage other conversion efforts and researchers to make their designs and implementation issues known, even if only on a limited basis, so that we can all jointly share in the progress toward true conversion.

References

1. Ramamoorthy, C.V., et al. "Software Engineering: Problems and Perspectives," *IEEE Computer,* Oct. 1984, pp. 191-209.
2. Jones, C. *Programmer Productivity.* McGraw-Hill, New York, N.Y., 1986.
3. Boehm, B., and T. Standish. "Software Technology in the 1990's: Using an Evolutionary Paradigm," *IEEE Computer,* Nov. 1983, pp. 30-37.
4. Manley, J. "Computer-Aided Software Engineering: Foundations for Software Factories," *Proceedings of CompCon,* Washington D.C., Sept. 1984, pp. 26-37.
5. Eliot, L., and W. Scacchi. "Towards a Knowledge-Based System Factory: Issues and Implementations," *IEEE Expert,* Winter 1986, pp. 51-58.
6. Tajima, D., and T. Matsubara, "Inside the Japanese Software Factory," *IEEE Computer,* Mar. 1984, pp. 34-43.
7. Leavitt, D. "Conversion no Longer a Trauma; Now Usually a Simple Migration," *Software News,* Sept. 1986, pp. 60-66.
8. Kaelbling, L. "FORADA: A FORTRAN to Ada Translator," *Ada Letters,* Sept. 1986, pp. vi.6-107—vi.6-108.
9. Aharonian, G. "Automated FORTRAN Conversion," *Proceedings of the First International Conference on Ada Programming Applications for the NASA Space Station,* June 1986, pp. D.1.3.1—D.1.3.13.
10. Slape, J., and P. Wallis, "Conversion of FORTRAN to Ada Using an Intermediate Tree Representation, *The Computer Journal,* Nov. 1983, pp. 344-353.
11. Wallis, P., "Automatic Language Conversion and its Place in the Transition to Ada," *Ada Letters,* Sept. 1985, pp. 275-284.
12. Ichbiah, et al. *Military Standard Ada Programming Language.* U.S. Government Printing Office, 1983.
13. X3J3 Committee. *American National Standard Programming Language FORTRAN.* American National Standards Institute, 1978.

We gratefully acknowledge the assistance of Nicholas Baker in the preparation of the FORTRAN-to-Ada converter.

Lance B. Eliot, Ph.D., is director of the Expert Systems Laboratory at the University of Southern California and a professor of Systems Science. He has published numerous articles and is currently interested in Ada, cognitive aspects of software development, artificial intelligence and expert systems, Ada and artificial intelligence, and related topics. Eliot is also a columnist for AI EXPERT *and holds the C.D.P., C.C.P., and C.S.P. certifications.*

Floyd L. Holliday is a professor of computer science at California State University, Fullerton. He served as the primary faculty member involved in the development of the FORTRAN-to-Ada converter described in this article. His interests are in Ada, software engineering, and artificial intelligence.


```
Converter segment from second pass

procedure emit_subprogram_preamble is
begin
  emit_new_line;
  emit_indentation; emit("with TEXT_IO;"); emit_new_line;
  emit_subprogram_withs(subprogram_table);
  emit_common_withs(common_table);
end emit_subprogram_preamble;

procedure emit_subprogram_specs is
  s: subprograms:=first_subprogram;
begin
  while s/=null loop
    if s.spelling/=main_subprogram.spelling then
      emit("with FORTRAN; use FORTRAN;"); emit_new_line;
      emit("       ");
      emit_subprogram_specification(s); emit(";"); emit_new_line;
    end if
    s := s.next;
  end loop;
end emit_subprogram_specs;

procedure emit_preamble is
begin
  indentation:= 0;
  emit("-- The main program is:"); emit_new_line;
  emit("with FORTRAN; use FORTRAN;"); emit_new_line;
  emit("       procedure ");
  emit_spelling(main_subprogram.spelling, false);
  emit(";"); emit_new_line; emit_new_line;
  emit("-- The procedure and functions are:"); emit_new_line;
  emit_subprogram_specs;
  emit_new_line;
  emit_common_packages(common_table);
end emit_preamble;
```

Listing 6.

```
Converter segment showing
internal debugging feature

    for i in 1..call_depth loop
      TEXT_IO.put(" ");
    end loop;
    TEXT_IO.put_line("into "&s);
    call_depth:= call_depth + 1;
  end if;
end into;
procedure from(s:string) is
begin
  if debugging then
    call_depth := call_depth-1;
    for i in 1..call_depth loop
      TEXT_IO.put(" ");
    end loop;
    TEXT_IO.put_line("from "&s);
  end if;
end from;
procedure note(s:string) is
begin
  if debugging then
    for i in 1..call_depth loop
      TEXT_IO.put(" ");
    end loop;
    TEXT_IO.put_line("note:"&s);
  end if;
end note;
procedure reset is
begin
  call_depth:= 0;
end reset;

procedure set_file(s:string) is
begin
 if s'length=0 or else s(1)=' ' then
   debugging:= false;
 else
   debugging:= true;
 end if;
end set_file;

end debug;

with TEXT_IO;
package body DEBUG is
  debug_file: string(1..100):=
    (others=>' ');
debugging: boolean:=false;
call_depth: integer:=0;

procedure into(s:string) is
begin
  if debugging then
```

Listing 7.

Artwork: Dwight Been

Discover the Difference

We're Programmer's Connection, your best one-stop source for quality programmer's development tools for IBM Personal Computers and compatibles. We invite you to compare us with the other dealers in our industry:

FREE Shipping. Shipping to U.S. customers is FREE via UPS Ground. If you want your order shipped via an express service, we'll only charge you the shipping carrier's standard rate with no special fees. Some dealers charge extra for shipping and then add rush charges for shipping via express services. Others may advertise "free" shipping but make up for it by charging extra handling fees.

Credit Cards. We'll charge your credit card only when we actually ship your order. Some dealers would charge your credit card at the time you place your order. This could leave you waiting for your shipment for weeks or months while they use your money interest-free.

Discounts. We discount all software products — even special order items. Every product in our advertised price list is shown with its list price and discounted price. We want you to know exactly how much you'll save on every product. We don't try to fool you by discounting some products and charging full retail for others.

Consistent Prices. We extend the same current prices to every customer regardless of where they see our ad. Some dealers vary prices in different ads and then ask you to mention which one you saw. This technique allows them to charge you the highest prices possible.

No Hidden Charges. The discount prices you see on the next two pages are all you pay. We don't charge extra for UPS Ground shipping, credit cards, COD orders, purchase orders, sales tax (except Ohio) or special handling (except for non-Canadian international orders).

Guarantees. We offer FREE 30-day no-risk return guarantees and 30-day evaluation periods on most of our products. Some dealers have no return options while others often charge restocking fees of 15% or more.

Quality Products. Our product line consists of hundreds of high quality software development tools specifically for IBM Personal Computers and compatibles. While some dealers try to carry every software product ever written, we carry only those that meet our very high standards for quality and value.

Latest Versions. The products we carry are the latest versions and come with the same manufacturer's technical support as if buying direct. While some dealers may participate in the software gray market, we're authorized to sell every product we carry.

Large Inventory. We have one of the largest inventories of programmer's development products in the industry. Most orders are shipped within 24 hours. And if we don't have a product in stock, we'll get it for you fast.

Meticulous Packaging. We'll give your shipment the extra protection needed to reach you in the best possible condition. First we'll protect your products from moisture by wrapping them in plastic. Then we'll insulate your box with high quality bubble-wrapping instead of the messy styrofoam chips that many other dealers use. Finally, we'll double-tape your box for extra strength.

Independence. Since we're not directly affiliated with any software publisher or developer, we can give you sound, unbiased advice. Unlike some other dealers who have a special interest in promoting only certain products, we'll give you an objective look at the products we carry.

Noncommissioned Staff. Our courteous sales staff is always ready to help you. And if you aren't sure about your needs, our knowledgeable technical staff can give you sound, objective advice. Because they are noncommissioned, you won't be pressured into making a purchase.

As you can see, we're not like the other dealers in this industry. Our customers keep coming back because we consistently provide the highest quality service and the lowest prices. So call us today and discover the difference for yourself.

PROGRAMMER'S CONNECTION
QUALITY INTEGRITY Service SUPPORT PRICE

	List	Ours

ai - expert systems

1st-CLASS *by Programs in Motion*	495	399
AutoIntelligence *by IntelligenceWare*	990	739
ExpertEDGE Advanced *by Human Edge*	2500	CALL
ExpertEDGE Professional *by Human Edge*	5000	CALL
Experteach II *by IntelligenceWare*	475	339
EXSYS *Development Software by EXSYS*	395	309
EXSYS Runtime System	600	469
Insight 1 *by Level Five Research*	95	75
Insight 2+ *by Level Five Research*	485	379
Intelligence/Compiler *by IntelligenceWare*	990	739
Logic-Line Series 1 *by Thunderstone*	90	85
Logic-Line Series 2 *by Thunderstone*	125	115
Logic-Line Series 3 *by Thunderstone*	150	139

ai - lisp language

GCLISP *Golden Common LISP by Gold Hill*	495	CALL
GCLISP 286 Developer *by Gold Hill*	1190	CALL
Microsoft LISP *Common LISP*	250	149
QNIAL *Combines LISP & APL by NIAL Systems*	375	339
Starter QNIAL *by NIAL Systems* *New*	99	89
TransLISP *from Solution Systems*	95	79
TransLISP PLUS *from Solution Systems*	195	169

ai - prolog language

APT *Active Prolog Tutor from Solution Systems*	65	55
Arity Combination Package	1095	979
Expert System Development Pkg	295	229
File Interchange Toolkit	50	44
PROLOG Compiler & Interpreter	650	569
Screen Design Toolkit	50	44
SQL Development Package	295	229
Arity PROLOG Interpreter	295	229
Arity Standard Prolog	95	77
LPA microPROLOG *All Varieties*	CALL	CALL
MPROLOG Language Primer *LOGICWARE*	50	45
MPROLOG P500 *by LOGICWARE*	495	395
MPROLOG P550 *by LOGICWARE*	220	175
Prolog-86 *from Solution Systems*	125	88
Prolog-86 Plus *from Solution Systems*	250	199
Turbo PROLOG *by Borland Intl*	100	63
Turbo PROLOG Toolbox *by Borland Intl*	100	64

ai - smalltalk language

Smalltalk/V *by Digitalk*	99	84
EGA/CGA Color Option	49	45
Goodies Diskette	49	45
Smalltalk/Comm	49	42

ai - texas instruments

PC Scheme *Lisp*	95	84
Personal Consultant Easy	495	435
Personal Consultant Plus	2950	2589
Personal Consultant Runtime	95	84

apl language

APL*PLUS/PC *by STSC*	595	424
APL*PLUS/PC Spreadsheet Mgr *by STSC*	195	139
APL*PLUS/PC Tools Vol 1 *by STSC*	295	199
APL*PLUS/PC Tools Vol 2 *by STSC*	85	58
ATLAS*GRAPHICS *by STSC* *New*	450	329
Financial/Statistical Library *by STSC*	275	189
Pocket APL *by STSC*	95	69
STATGRAPHICS *by STSC*	795	579

assembly language

386 ASM/LINK *Cross Asm by Phar Lap*	495	389
8088 Assembler *w/Z-80 Translator by 2500 AD*	100	89
ASMLIB *Function Library by BC Assoc*	149	125
asmTREE *B-Tree Dev System by BC Assoc*	395	339
Cross Assemblers *Various by 2500 AD*	CALL	CALL
Microsoft Macro Assembler	150	93
Norton Utilities *by Peter Norton*	100	CALL
Norton Utilities (Advanced) *New*	150	99
screenplay *by Flexus*	100	79
Turbo EDITASM *by Speedware*	99	84
Uniware Cross Assemblers *Various by SDS*	CALL	CALL
Visible Computer: 8088 *by Software Masters*	80	64

basic language

87 Software Pak *by Hauppauge*	180	149
EXIM Services Toolkit *by EXIM*	50	45
Finally *by Komputerwerks*	99	85
Inside Track *from Micro Help*	65	51
MACH 2 *by Micro Help*	69	55
MicroHelp Utilities *by MicroHelp* *New*	59	49
Microsoft QuickBASIC Compiler *New version*	99	63
Peeks 'n Pokes *from MicroHelp*	45	37
Professional BASIC *by Morgan*	99	68
8087 Math Support	50	42
Quick-Tools *by BC Associates* *New*	130	109
QuickPak *by Crescent Software*	69	59
Scientific Subroutine Library *by Peerless*	125	99
Screen Sculptor *by Software Bottling*	125	91
Stay-Res *by MicroHelp*	95	73
True Basic *w/Run-time by True Basic*	245	179
True Basic	150	97
Run-time Module	150	97
Various Utilities	50	41
Turbo BASIC Compiler *by Borland Intl*	100	64

blaise products

ASYNCH MANAGER *Specify C or Pascal*	175	135
C TOOLS PLUS	175	135
EXEC *Program Chainer*	95	75
LIGHT TOOLS *for Datalight C*	100	79
PASCAL TOOLS	125	99
PASCAL TOOLS 2	100	79
PASCAL TOOLS & TOOLS 2	175	135
RUNOFF *Text Formatter*	50	45
TURBO ASYNCH PLUS	100	79
TURBO C TOOLS *New*	129	CALL
TURBO POWER TOOLS PLUS	100	79
VIEW MANAGER *Specify C or Pascal*	275	199

borland products

EUREKA *Equation Solver*	100	64
Reflex & Reflex Workshop	200	128
Reflex *Data Base System*	150	89
Reflex Workshop	70	45
Sidekick & Traveling Sidekick	125	85
Sidekick	85	57
Traveling Sidekick	70	45
Superkey	100	64
Turbo BASIC Compiler	100	64
Turbo C Compiler	100	64
Turbo Database Toolbox	70	41
Turbo Editor Toolbox	70	41
Turbo Gameworks Toolbox	70	41
Turbo Graphix Toolbox	70	41
Turbo Jumbo Pack	300	219
Turbo Lightning	100	64
Turbo PASCAL Numerical Methods Toolbox	100	64
Turbo PASCAL and Tutor	125	85
Turbo PASCAL	100	64
Turbo Tutor	40	24
Turbo PROLOG Compiler	100	63
Turbo PROLOG Toolbox	100	64
Word Wizard	70	47
Word Wizard and Turbo Lightning	150	94

c++

C++ *by Guidelines w/version 1.1 kernel*	195	172
PforCe++ *Function Library by Phoenix*	395	225

c compilers

C86PLUS *by Computer Innovations*	497	359
Datalight C Compiler *by Datalight*	60	43
Datalight Developer Kit	99	74
Datalight Optimum-C	139	99
DeSmet C *w/Debugger & Large case* .. *NewVersion*	209	184
DeSmet C *w/Debugger only* ..... *New Version*	159	138
Eco-C *with Graphics by Ecosoft*	125	83
Lattice C Compiler *vers. 3.2 from Lattice*	500	265
Mark Williams Let's C Combo Pack .. *New Version*	125	99
Let's C *Compiler* ..... *New Version*	75	54
csd *Source Level Debugger* ..... *New Version*	75	54
Microsoft C *with CodeView*	450	269
Turbo C Compiler *by Borland*	100	64
Uniware 68000/10/20 Cross Compiler *by SDS*	CALL	CALL

c interpreters

C-terp *by Gimpel, Specify compiler*	300	219
C Trainer *with Book by Catalytix*	122	87
Instant C *by Rational Systems*	500	369
Introducing C *by Computer Innovations*	125	99
Run/C *by Age of Reason*	120	79
Run/C Professional *by Age of Reason*	250	157

c utilities

c-tree & r-tree Combo *by FairCom*	650	529
c-tree *ISAM File Manager*	395	329
r-tree *Report Generator*	295	249
C Windows *by Syscom*	100	85
C Wings *by Syscom*	50	43
dBx *dBASE to C Translator by Desktop Al*	350	299
EASY SCREEN *by Retail Mgmt Systems* ..... *New*	225	199
Flash-up Windows *by Software Bottling*	90	78
GraphiC *Color version by Sci Endeavors*	350	282
GRAFLIB *by Sutrasoft*	175	159
HALO *Graphics by Media Cybernetics*	300	205
HALO Development Pkg *for Microsoft*	595	389
The HAMMER *by OES Systems*	195	129
PANEL *Forms Management by Roundhill*	295	CALL
PANEL Plus *by Roundhill*	495	CALL
PC Lint *by Gimpel Software*	139	99
PLOTHI *by Sutrasoft*	175	159
PLOTHP *by Sutrasoft*	175	159
Professional C Windows *by Washburn*	89	79
Scientific Subroutine Library *by Peerless*	175	128
screenplay *for C by Flexus*	150	129
Vitamin C *by Creative Programming*	225	158
VC Screen *Forms Designer*	100	79
Zview *by Data Mgmt Consultants*	245	139

cobol language

COBOLspll *by Flexus*	395	329
FPLIB *for Realia COBOL by BC Associates*	149	129
Micro Focus COBOL *See Micro Focus Section*		
Microsoft COBOL *See Microsoft Section*		
Realia COBOL with RealMENU ..... *New*	1145	899
Realia COBOL	995	783
RealCICS	995	783
RM/COBOL *by Ryan-McFarland*	950	CALL
RM/COBOL 85 *by Ryan-McFarland*	1250	CALL
screenplay *for COBOL by Flexus*	175	139

css products

PC/SPELL *by Custom Software Systems* ..... *New*	49	45
PC/TOOLS ..... *New*	49	45
PC/TOP ..... *New*	49	45
PC/VI	149	99

debuggers & profilers

386 DEBUG *Cross Debugger by Phar Lap*	195	129
Advanced Trace-86 *by Morgan Computing*	175	115
Codesifter *Profiler by David Smith*	119	85
Codesmith-86 *by Visual Age*	145	98
DSD87 *by Soft Advances*	125	79
MiniProbe *by Atron*	395	369
Periscope I *with Board by Periscope*	345	277
Periscope II *with NMI Breakout Switch*	175	139
Periscope II-X *Software only*	145	105
Periscope III *8 MHz version*	995	825
Periscope III *10 MHz version*	1095	899
The PROFILER *with Source Code by DWB*	125	89
TURBOsmith *Source debugger for Turbo Pascal*	69	59
The WATCHER *Profiler by Stony Brook*	60	51

disk utilities

Back-It *by Gazelle Systems* ..... *New*	100	89
Disk Optimizer *by Softlogic Systems*	60	55
FASTBACK *by 5th Generation Systems*	179	129
XenoCopy-PC *by XenoSoft* ..... *New*	80	69

dos utilities

Command Plus *by ESP Software*	80	69
FANSI-CONSOLE *by Hersey Micro*	75	62
Norton Commander *by Peter Norton*	75	CALL
OPAL *Shell Language by Software Factory* ... *New*	99	89
OPART *Screen Painter* ..... *New*	75	67
Q-DOS II *by Gazelle Systems* ..... *New*	70	59
Scroll & Recall *by Opt-Tech Data*	69	59
Taskview *by Sunny Hill Software*	80	55

essential products

Turbo C versions available.

C Essentials *by Essential Software*	100	75
C Utility Library	185	119
Essential Comm Library *with Debugger*	250	189
Essential Comm Library *Software Only*	185	125
Breakout Debugger Only *Any language*	125	89
Essential Graphics	250	183

forth language

CFORTH *Native Code Compiler by LMI*	300	229
Forth/83 Metacompiler *Specify Target*	750	599
PC/Forth *by Laboratory Microsystems*	150	109
PC/Forth+ *by Laboratory Microsystems*	250	199
Advanced Color Graphics Support	100	74
Enhanced Graphics Support	200	148
Intel 8087 Support	100	74
Interactive Symbolic Debugger	100	74
Native Code Optimizer	200	148
Software Floating Point	100	74
UR/Forth *by LMI*	350	279
Object Module Libraries	500	395

fortran language

50 MORE: FORTRAN *by Peerless Engr*	125	95
ACS Time Series *by Alpha Computer Service*	495	389
Essential Graphics *by Essential Software*	250	183
For-Winds *by Alpha Computer Service*	90	69
Forlib-Plus *by Alpha Computer Service*	70	44
FORTLIB *by Sutrasoft*	125	109
FORTRAN Addendum *by Impulse Engr*	95	85
FORTRAN Addenda *by Impulse Engr*	165	138
GRAFLIB *by Sutrasoft*	175	159
HALO *Graphics by Media Cybernetics*	300	205
I/O PRO *by MEF Environmental*	149	129
Microcompatibles Combo Package	240	215
Grafmatic	135	117
Plotmatic	135	117
Microsoft FORTRAN *w/CodeView*	450	269
No Limit *by MEF Environmental*	129	109
Numerical Analyst *by MAGUS*	295	249
PANEL *by Roundhill Computer Systems*	295	CALL
PLOTHI *by Sutrasoft*	175	159
PLOTHP *by Sutrasoft*	175	159
RM/FORTRAN *by Ryan-McFarland*	595	CALL
RTC PLUS *Fortran to C by Cobalt Blue*	325	289
Scientific Subroutine Lib *by PSI/Systems*	175	128
Statistician *by Alpha Computer Service*	295	235
Statlib.GL: *by PSI/Systems*	295	239
Statlib.TSF: *by PSI/Systems*	295	239
Strings & Things *by Alpha Computer Service*	70	45

greenleaf products

Greenleaf Comm Library	185	125
Greenleaf Data Windows	225	155
with Source Code	450	289
Greenleaf Functions	185	125

help utilities

HELP/Control *by MDS*	125	99
On-line Help *from Opt-Tech*	149	99
SoftScreen/HELP *by Dialectic Systems*	195	149

lattice products

Lattice C Compiler *ver 3.2 from Lattice*	500	265
with Library Source Code	900	495
C Cross Reference Generator	50	37
with Source Code	200	139
C-Food Smorgasbord *Function Library*	150	95
with Source Code	300	179
C-Sprite *Source Level Debugger*	175	119
Curses *Screen Manager*	125	85
with Source Code	250	169
dBC *Specify dBC II or dBC III*	250	169
with Source Code	500	356
dBC III Plus	750	594
with Source Code	1500	1184
LMK *Make Facility*	195	138
RPG II Combo *All three items below*	1100	875
RPG II Compiler *No Royalties*	750	625
RPG II SEU *Screen Entry Utility*	250	199
RPG II Sort/Merge	250	199
RPG II Screen Design Aid Utility	350	309
SecretDisk *File Encryption Utility*	120	88
SideTalk *Resident Communications*	120	88
SSP/PC *Scientific Subroutine Library*	350	269
Text Management Utilities	120	88
TopView Toolbasket *Function Library*	250	178
with Source Code	500	356

metagraphics products

LightWINDOW/C *for Datalight C*	95	79
Font WINDOWS	95	79
Font WINDOWS/Plus	275	229
MetaWINDOW *No Royalties*	195	159
MetaWINDOW/PLUS	275	229
TurboWINDOW/C *for Turbo C*	95	79
TurboWINDOW/Pascal *for Turbo Pascal*	95	79

micro focus products

Micro Focus Level II COBOL *w/Animator* .... *New*	495	395
Level II COBOL	349	279
Level II Animator	195	155
Micro Focus Level II COBOL/ET for UNIX ... *New*	CALL	CALL
Micro Focus Personal COBOL	149	119
Micro Focus Professional COBOL	2000	1595
Micro Focus VS COBOL/XENIX ..... *New*	1495	1195

Micro Focus Support Products:		
COBOL/IQ *Ad hoc Report Writer* ..... *New*	495	395
COBOL/IQ *for DOS 3.X Networks* ..... *New*	995	795
FORMS-2	295	235
SOURCEWRITER	995	795

microport products

System V/AT *by Microport Systems*	549	465
Runtime System *(Operating System)*	199	169
Software Development System	249	209
Text Preparation System	199	169
Unlimited License Kit	249	209

microsoft products

Microsoft BASIC Compiler *for XENIX* ..... *New*	695	419
Microsoft BASIC Interpreter *for XENIX*	350	209
Microsoft C Compiler *with CodeView*	450	269
Microsoft COBOL Compiler *with COBOL Tools*	700	429
for XENIX	995	609
Microsoft FORTRAN *Optimizing Compiler/CodeView*	450	269
Microsoft FORTRAN *for XENIX*	695	419
Microsoft Learning DOS	50	36
Microsoft LISP *Common LISP*	250	149
Microsoft MACH 10 *with Mouse & Windows*	549	369
Microsoft MACH 10 *Board only*	399	279
Microsoft Macro Assembler	150	93
Microsoft Mouse *Bus Version*	175	114
Microsoft Mouse *Serial Version*	195	124
Microsoft muMath *Includes muSIMP*	300	179
Microsoft Pascal Compiler	300	179
for XENIX	695	419
Microsoft QuickBASIC Compiler .... *New version*	99	63
Microsoft Sort	195	125
Microsoft Windows	99	63
Microsoft Windows Development Kit	500	299
Microsoft Word ..... *Special Price thru July*	450	225

modula-2 language

MODULA-2 Apprentice Pkg by LOGITECH	99	79
MODULA-2 Magic Pkg by LOGITECH	99	79
MODULA-2 ROM Pkg & Cross RT Debugger	299	239
MODULA-2 Window Pkg by LOGITECH	49	39
MODULA-2 Wizard's Pkg by LOGITECH	199	159
REPERTOIRE *for MODULA-2 by PMI*	89	75
Object Code Only	19	15

mouse products

LOGIMOUSE BUS *with PLUS Pkg by LOGITECH*	119	98
with PLUS & PC Paintbrush	149	119
with PLUS & CAD Software	189	153
with PLUS & CAD & Paint	219	179
LOGIMOUSE C7 *with PLUS Pkg, Specify Connector*	119	98
with PLUS & PC Paintbrush	149	119
with PLUS & CAD Software	189	153
with PLUS & CAD & Paint	219	179
Microsoft Mouse *Bus Version*	175	114
Microsoft Mouse *Serial Version*	195	124

other languages

CCS MUMPS *Single-User by MGlobal*	60	50
CCS MUMPS *Single-User Multi-Tasking*	150	129
CCS MUMPS *Multi-User*	450	359
Janus/ADA C Pak *by R&R Software*	95	84
Janus/ADA D Pak *by R&R Software*	900	769
Janus/ADA ED Pak *by R&R Software*	395	349
Marshal Pascal *by Marshal Language Systems*	189	155
Pascal-2 *by Oregon Software* ..... *New*	395	325
Personal REXX *by Mansfield Software*	125	99
SNOBOL4+ *by Catspaw*	95	80

other products

Dan Bricklin's Demo Pgm *by Software Garden*	75	57
Dan Bricklin's Demo Tutorial	50	45
Informix *All Varieties by Informix*	CALL	CALL
Instant Replay *by Nostradamus* ..... *New Version*	150	CALL
MKS Toolkit *with vi Editor by MKS*	139	114
MicroTEX *Typesetting from Addison Wesley* .. *New*	295	CALL
Net-Tools *by BC Associates*	149	129
OPT-Tech Sort *by Opt-Tech Data Proc*	149	99
PC/TOOLS *by Custom Software* ..... *New*	49	45
Screen Machine *by MicroHelp*	79	59
VTEK *Term Emulator by Sci Endeavors*	150	129

phoenix products

Pasm86 *Macro Assembler version 2.0*	195	108
Pdisk *Hard Disk & Backup Utility*	145	89
Pfantasy Pac *Phoenix Combo*	995	599
Pfinish *Execution Profiler*	395	209
Pfix86plus *Symbolic Debugger*	395	209
PforCe *Specify C Compiler*	395	209
PforCe++ *Specify C Compiler and C++*	395	209
Plink86plus *Overlay Linker*	495	279
Pmaker *Make Utility*	125	78
Pmate *Macro Text Editor*	195	108
Pre-C *Lint Utility*	295	154
Ptel *Binary File Transfer Program*	195	108

polytron products

PolyBoost *The Software Accelerator*	80	64
PolyDesk III ..... *New*	99	72
PolyDesk III Archivist ..... *New*	50	42
PolyDesk III Cryptographer ..... *New*	50	42
PolyDesk III Talk ..... *New*	70	52
PolyLibrarian *Library Manager*	99	73
PolyLibrarian II *Library Manager*	149	109
PolyMake *UNIX-like Make Facility*	149	109
PolyShell	149	109
Polytron C Beautifier	50	42
Polytron C Library I	99	72
Polytron PowerCom *Communications*	139	105
PolyXREF *Complete Cross Ref Utility*	219	169
PolyXREF *One language only*	129	99
PVCS Network *Version Control System* ..... *New*	1000	CALL
PVCS Corporate	395	309
PVCS Personal	149	109
PVMFM *Polytron Virtual Memory File Mgr* .... *New*	199	155

program mgmt utilities

Compact Source Print *by Aldebaran*	55	44
Interactive EASYFLOW *by Haventree*	150	125
PrintQ *by Software Directions*	89	84
Quilt Computing Combo Package ..... *New Version*	250	199
QMake *Program Rebuild Utility*	99	79
SRMS *Software Revision Mgmt Sys* ... *New Version*	185	159
Source Print *by Aldebaran Labs*	75	59
TLIB *by Burton Systems Software*	100	89
Tree Diagrammer *by Aldebaran Labs*	55	49

raima products

dbQUERY *Single-User Query Utility*	195	129
Single-User with Source Code	495	389
Multi-User	495	389
Multi-User with Source Code	990	699
dbVISTA *Single-User DBMS*	195	129
Single-User with Source Code	495	389
Multi-User	495	389
Multi-User with Source Code	990	699

sco products

Complete XENIX System V *by SCO*	1295	994
Development System	595	499
Operating System *Specify XT or AT*	595	499
Text Processing Package	195	144
Lyrix *by SCO*	595	449
XENIX-NET *by SCO*	595	495
SCO Professional *1-2-3 Workalike for XENIX*	795	595

softcraft products

Btrieve *ISAM Mgr with No Royalties*	245	184
Xtrieve *Query Utility*	245	184
Report Option *for Xtrieve*	145	99
Btrieve/N *for Networks*	595	454
Xtrieve/N	595	454
Report Option/N *for Xtrieve/N*	345	269

text editors

Brief & dBrief Combo *from Solution Systems*	250	CALL
Brief ..... *New version*	195	CALL
dBrief *Customizes Brief for dBASE III*	95	CALL
Epsilon *Emacs-like editor by Lugaru*	195	147
KEDIT *by Mansfield Software*	125	98
Micro/SPF *by Phaser Systems*	175	139
Microsoft Word ..... *Special Price thru July*	450	225
PC/VI *by Custom Software Systems*	149	99
SPF/PC *by Command Technology Corp*	CALL	CALL
Vedit *by CompuView*	150	98
Vedit Plus *by CompuView*	185	128

turbo pascal utilities

ALICE *Interpreter by Software Channels*	95	66
DOS/BIOS & Mouse Tools *by Quinn-Curtis*	75	67
Flash-up Windows *by Software Bottling*	90	78
MACH 2 *for Turbo Pascal by Micro Help*	69	55
MetraByte D/A Tools *by Quinn-Curtis*	100	89
Science & Engrg Tools *by Quinn-Curtis*	75	67
Screen Sculptor *by Software Bottling*	125	91
screenplay *for Turbo Pascal by Flexus*	100	79
Speed Screen *by Software Bottling*	35	32
System Builder *by Royal American*	130	115
IMPEX *Query Utility*	75	69
Report Builder	75	69
TDebugPLUS *by TurboPower Software*	60	49
Turbo EXTENDER *by TurboPower Software*	85	64
Turbo OPTIMIZER *by TurboPower* ..... *New*	75	65
Turbo OPTIMIZER *with Source Code* ..... *New*	125	108
Turbo Professional *by Sunny Hill*	70	45
TurboHALO *from IMSI*	129	98
TurboPower Utilities *by TurboPower*	95	78
TurboRef *by Gracon Services*	50	45
TURBOsmith *Source Debugger by Visual Age*	69	59
Universal Graphics Library *by Quinn-Curtis* .... *New*	150	119

wendin products

Operating System Toolbox ..... *Rebate Offer*	99	75
PCNX *Operating system* ..... *Rebate Offer*	99	75
PCVMS *Similar to VAX/VMS*	99	75
XTC *Text Editor w/Pascal source*	99	75

xenix/unix products

Btrieve *ISAM File Mgr by SoftCraft*	595	454
C-terp *by Gimpel*	498	379
c-tree *ISAM Mgr by FairCom*	395	329
dBx *with Library Source by Desktop AI*	550	489
DOSIX *Console Version by Data Basics*	399	349
DOSIX *User Version by Data Basics*	199	179
Informix *All Varieties by Informix*	CALL	CALL
Micro Focus Products *See Micro Focus Section*		
Microport Products *See Microport Section*		
Microsoft Products *See Microsoft Section*		
PANEL Plus *by Roundhill Computer Systems*	495	CALL
REAL-TOOLS *Binary Version by PCT*	149	89
Library Source Version	399	289
Complete Source Version	499	369
RM/COBOL *by Ryan-McFarland*	1250	CALL
RM/FORTRAN *by Ryan-McFarland*	750	CALL
SCO Products *See SCO Section*		

LOWEST PRICES

Due to printing lead times, some of our current prices may differ from those shown here. Call for latest pricing.

FREE SHIPPING

Orders within the USA (including Alaska & Hawaii) are shipped FREE via UPS. Express shipping is available at the shipping carrier's standard rate with no rush fees or handling charges. To avoid delays when ordering by mail, please call first to determine the exact cost of express shipping.

CREDIT CARDS

VISA, MasterCard and Discover Card are accepted at no extra cost. Your card is charged when your order is shipped. Mail orders please include credit card expiration date and authorized signature.

CODs AND POs

CODs and Purchase Orders are accepted at no extra cost. POs with net 30-day terms are available to qualified US accounts only.

SALES TAX

Orders outside of Ohio are not charged state sales tax. Ohio customers please add 6% Ohio tax or provide proof of tax-exemption.

INTERNATIONAL ORDERS

Shipping charges for International and Canadian orders are based on the shipping carrier's standard rate. Since rates vary between carriers, please call or write for the exact cost. International orders (except Canada), please include an additional $10 for export preparation. All payments must be made with US funds drawn on a US bank. Please include your telephone number when ordering by mail. Due to government regulations, we cannot ship to all countries.

VOLUME ORDERS

Volume orders may qualify for additional discounts. Call us for special pricing.

SOUND ADVICE

Our knowledgeable technical staff can answer technical questions, assist in comparing products and send you detailed product information tailored to your needs.

30-DAY GUARANTEE

Most of our products (excluding books) come with a 30-day documentation evaluation period or a 30-day return guarantee. Please note that some manufacturers restrict us from offering guarantees on their products. Call for more information.

MAIL ORDERS

Please include your telephone number on all mail orders. Be sure to specify computer, operating system and any applicable compiler or hardware interface(s). Send mail orders to:

Programmer's Connection
Order Processing Department
136 Sunnyside Street
Hartville, OH 44632

CALL TOLL FREE

USA **800-336-1166**
CANADA **800-225-1166**
OHIO & ALASKA (Collect) .. **216-877-3781**

TELEX **9102406879**
EASYLINK **62806530**
INTERNATIONAL **216-877-3781**
CUSTOMER SERVICE .. **216-877-1110**

Hours: Weekdays 8:30 AM to 8:00 PM EST.

Call or write for our FREE comprehensive price guide.

©Copyright Programmer's Connection, Inc., 1987.
All Rights Reserved.

programmer's connection

CIRCLE 34 ON READER SERVICE CARD

Arity/Prolog. Not for the meek. But Arity/Prolog® is for you...if you hunger for the power to create sophisticated programs right on the IBM® PC. Everything from weather prediction and process control systems to database design and medical diagnostic software.

It's for you if you're already building programs on the PC and now want to do things you've never done anywhere but on a VAX™ or mainframe. And it's for you if you've read all about the new programming technologies and are ready to start coding.

Whatever your need, Arity/Prolog offers a total development environment including Arity/SQL and Arity/Expert. It even has an open architecture so you can work with other languages and software packages.

So if you're ready to boldly face the future of programming now, write for more information and an Arity power mask: **Arity**, 30 Domino Dr., Concord, MA 01742. Or call **1-800-PC-Arity** (in MA 617-371-1243).

CIRCLE 35 ON READER SERVICE CARD

Multiway Branching in C

By Alan Filipski

Although the simple *if-then-else* statement provided by most programming languages suffices as a decision control structure for toy textbook problems or theoretical analyses, practical programs often make good use of multiway decision structures that consider more than two alternatives at a time.

An interactive program, for example, is frequently structured as a loop that first reads a command and then takes a multiway branch to decide upon an action. An assembler or disassembler often has many multiway branches for the decoding of assembler mnemonics or op codes. In addition, drivers often have to decode control bytes from external hardware by using such branches.

Multiway branches can be done two ways in the C programming language: with the switch statement or with indexed procedure invocation. The first is a control structure provided by the language; the second is a programming technique allowed by the language.

This article first discusses the switch control structure and how it is handled by typical C compilers. Then the less common technique of indexed procedure invocations is described as an occasionally useful alternative.

Switching in C

In C, the switch control structure for multiway decision making takes the following form:

```
switch ( e ) {
case c1:
   s1
case c2:
   s2
   . . .
   . . .
   . . .
case cn:
   sn
default:
   s
}
```

where *e* is an integer-valued expression, *c1, c2, ..., cn* are constant integer-valued expressions, and *s, s1, s2, ..., sn* represent sequences of statements.

The values of *c1, c2, ..., cn* must all be different. When the switch statement is executed, *e* is evaluated. If its value is equal to one of the constant expressions *ci,* control is transferred to the first statement following the label *case ci*; if its value does not equal any *ci,* control is transferred to the first statement following the special label *default.*

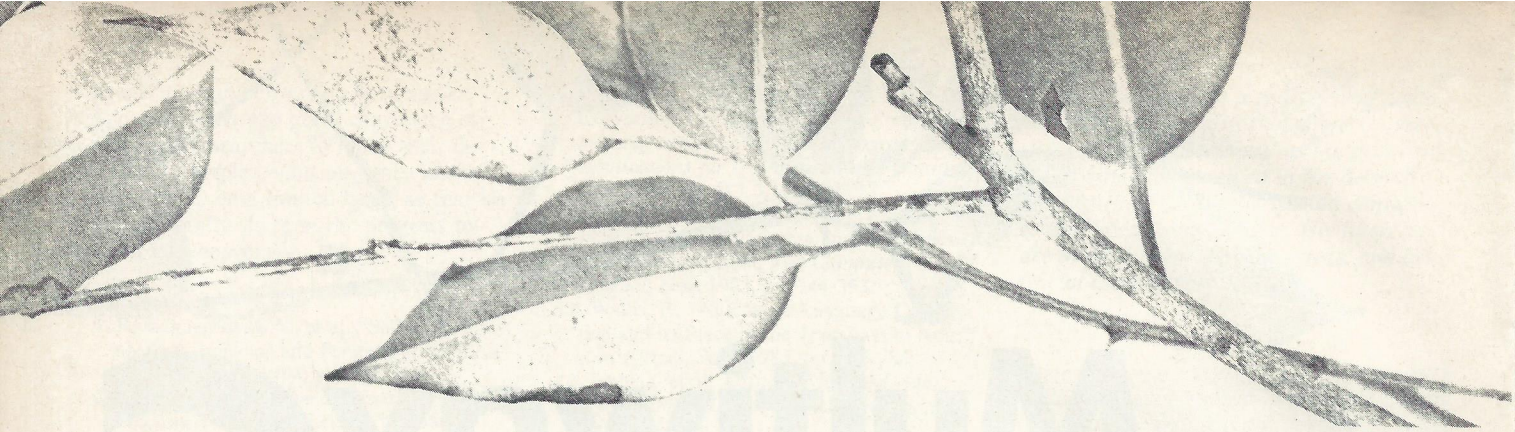
Once control is transferred inside the switch, execution proceeds through all following statements within the switch without regard for subsequent case labels. This situation is usually not desirable, so it is customary to add break statements to jump out of the switch before executing the statements corresponding to the next case:

```
switch ( e ) {
case c1:
   s1
   break;
case c2:
   s2
   break;
   . . .
   . . .
   . . .
case cn:
   sn
   break;
default:
   s
   break;
}
```

To the C programmer, executing the switch appears to consist of evaluating the integer expression and then immediately transferring control to the statement following the appropriate case label. This is an oversimplification. To use the switch intelligently in time-critical applications, it is important to understand how a switch is compiled and what actually happens during execution of the switch.

Most C compilers will compile a switch in different ways, depending upon the number and values of the case constants. For example, consider a case in which the case constants are the consecutive integers 0, 1, 2, and 3 (in all the examples, we will assume there is a break statement after the code for each case):

```
switch ( e ) {
case 0:
   <code for case 0>
   break;
```



```
case 1:
   <code for case 1>
   break;
case 2:
   <code for case 2>
   break;
case 3:
   <code for case 3>
   break;
default:
   <code for default case>
   break;
}
```

A typical C compiler translates this as:

```
   <code to evaluate expression e and
   place result in register, say, r0>
        jump to LA
L0      <code for case 0>
        jump to LB
L1      <code for case 1>
        jump to LB
L2      <code for case 2>
        jump to LB
L3      <code for case 3>
        jump to LB
LD      <code for default case>
        jump to LB
LA      if (r0>3 or r0<0) jump to LD
        jump to address at JTABLE+(4*r0)
JTABLE  L0
        L1
        L2
        L3
LB
```

We are assuming a byte-addressable machine and four-byte addresses, so that *JTABLE+(4*0)* contains *L0*, *JTABLE+(4*1)* contains *L1*, etc.

Listing 1A gives a concrete example of a C switch; Listing 1B gives its translation into Motorola 68010 assembly code (UNIX-style mnemonics). This approach to translation is fairly efficient and straightforward except for one jump (the jump to *LA* in the previous example), which is logically superfluous. If we were coding this by hand, we would move the code:

```
LA      if (r0>3 or r0<0) jump to LD
        jump to address at JTABLE+(4*r0)
JTABLE  L0
        L1
        L2
        L3
```

up to where the jump to *LA* is located and delete the jump.

For a compiler, however, it is easier to defer writing the jump table until all cases have been seen. The extra time taken by the jump is usually not important. The method given may be easily extended by the compiler designer to an offset range of case values by modifying the jump table or the calculation that indexes into the jump table.

For example, if the case values were 500, 501, 503, and 503, the indirect jump would be:

```
jump to address at JTABLE+
  (4*(r0-500))
```

But suppose the switch to be compiled is:

```
switch ( e ) {
case 0:
   <code for case 0>
   break;
case 1:
   <code for case 1>
   break;
case 2:
   <code for case 2>
   break;
case 1000:
   <code for case 1000>
   break;
default:
   <code for default case>
   break;
}
```

```
C program fragment--
dense switch

switch ( eval() )  {

case 0:
   proc0();
   break;

case 1:
   proc1();
   break;

case 2:
   proc2();
   break;

case 3:
   proc3();
   break;

default:
   procd();
   break;
}
```

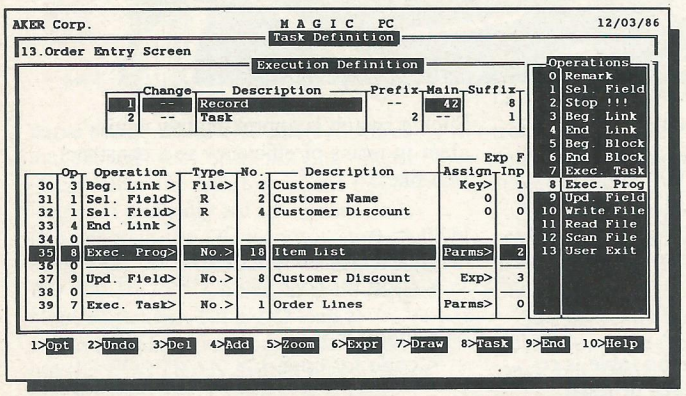
Listing 1A.

```
Equivalent M68010 code--
dense switch
        jsr     eval
        br      L%39
L%40:
        jsr     proc0
        br      L%38
L%41:
        jsr     proc1
        br      L%38
L%42:
        jsr     proc2
        br      L%38
L%43:
        jsr     proc3
        br      L%38
L%44:
        jsr     procd
        br      L%38
L%39:
        mov.l   &3,%d1
        cmp.l   %d0,%d1
        bhi     L%44
                add.w   %d0,%d0
        mov.w   10(%pc,%d0.w),%d0
        jmp     6(%pc,%d0.w)
        swbeg   &4
L%45:
        short   L%40-L%45
        short   L%41-L%45
        short   L%42-L%45
        short   L%43-L%45
L%38:
L%37:
```

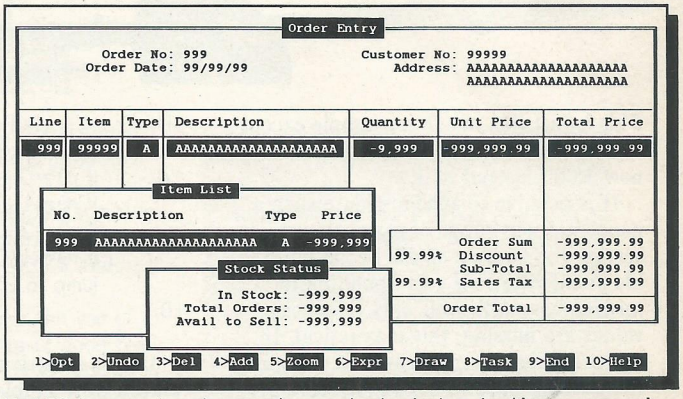
Listing 1B.

Announcing Magic PC – the first breakthrough for database applications developers in over 20 years:

Now you can develop professional applications 1000% faster than your 4GL or DBMS, totally free from programming, commands and syntax!



A Magic PC program looks as simple as this. To design an application you quickly fill-in menu-driven decision tables **without having to write a single line of code.** For example, just by highlighting the Execute Program operation on this screen and also highlighting the Item List program in the Program Menu, you tell Magic PC to pop-up the Item List window shown in the adjacent screen, when the end-user hits the Zoom key.



Magic PC gives your end-user the power to harness and retrieve data instantly, **without any commands or syntax** because at runtime you already have built-in options to Add, Delete, Modify, Query and get on-the-spot ad-hoc information simply by highlighting selections from menus. Dats validation, security and error-checking are done automatically for you by Magic PC without programming.

Who needs another DBMS?

At last, Magic PC gives you the ultimate applications design tool, far ahead of 4GL's, DBMS and Application Generators.

Magic PC breaks through the language barrier with the revolutionary Un-Language concept:

NO PROGRAMMING, COMMANDS OR SYNTAX!

Free yourself from your programming language

Magic PC makes you, the professional, completely free from the drudgery of procedural programming. No more cryptic commands, syntax or unforgiving procedural structures, because Magic PC does all the programming automatically. There's your competitive edge. The rest is up to you . . .

The Professional Choice

Already an international success, Magic PC is a profit maker and career booster for DP Consultants, System Integrators, VARs, MIS professionals, System Analysts, Programmer Analysts and Software Engineers. If you design PC applications professionally, you can't afford not to Un-Language now.

IBM France: "IBM encourages this introduction and can not help but salute such evolution . . ."

Israeli Air Force: "We were convinced that it was not possible to have a design tool powerful enough to implement real-life applications without a programming language. Magic PC changed our mind . . ."

Jeff Duntemann, PC Tech Journal: "It's probably the best integrated database applications and screen generator that I have ever seen . . .very smooth system, and smoothness comes at a premium these days . . ."

The Magic PC Secret

You're so much more productive with Magic PC because there is **absolutely no programming** to slow you down. You design a Magic PC application by simply filling-in the **Data Dictionary Tables** (Files, Fields, Keys) and the **Task Description Tables** (Operations and Expressions).

Only 13 design **Operations** harness the power of Magic PC. Operations are specific enough to eliminate the need for tiresome syntax, yet elastic enough to produce robust custom applications. Use the Operations to describe **what** you want and Magic PC makes it happen. It's that simple.

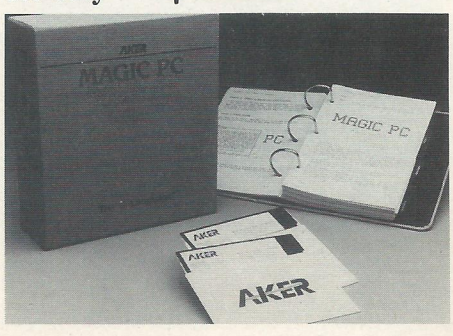
Make Task nesting power available with a single **Execute Task** Operation. This powerful instruction triggers Magic PC to execute and display additional tasks or even external applications through **Window Zooms.** The 3-dimensional effect of Window Zooming lets you probe deep into your application through nested windows and manipulate the data underneath.

You describe a Magic PC Task or Program (composite Tasks) by filling your system analysis flow into the Task Description Tables. Choose the participating Data View, and Magic PC executes your desired Operations. You interface with the Tables by highlighting your selections from pop-up menu-driven windows. There's nothing to edit except your headings.

You're not confined to any particular design sequence as you are with most procedural languages. You can enter and change any Table spontaneously, on the fly, as ideas come to mind and Magic PC automatically maintains the application integrity.

A **Magic Inference Engine** automatically orchestrates your Task Description Tables into a single file of internal **Knowledge Base Rules** for optimum, bug-free performance. Knowledge Base Rules are executed by the **Magic Run** engine for stand-alone runtime operation, or by the **Magic Lan** engine for unrestricted Novell network sharing. You're free to design the Knowledge Base without worrying about the internal structure.

Discover fast, language-free programming at no risk for only $19.95



See for yourself how fast you can program language-free applications with our low-cost limited offer.

You'll get the full Magic PC software unprotected and limited to 100 records and 450 page documentation complete with a **free** Order Entry sample application. You'll also get our **free** telephone support for 90 days!

And your $19.95 will be credited towards the full $695 Magic PC purchase price. Even if you don't buy Magic PC right away, keep your $19.95 Magic PC Trial as your application prototyping tool at this bargain price.

Our No-Risk Guarantee!

You have our no-risk 30-day money-back guarantee: if you're not completely satisfied for any reason, even Magic PC Trial for $19.95, send it back for a refund.

Order now while supply lasts

Call this toll free number now with your Visa, MasterCard or American Express for immediate delivery, or send the Order Coupon below today to Aker.

1-800-345-MAGIC
in CA call 714-250-1718

VISA MasterCard AMERICAN EXPRESS

Yes, please rush me:

☐ Magic PC Trial	$ 19.95
☐ Magic PC	$695.00
Add shipping	$ 5.00
In CA 6% tax	$_____
Prices valid in US only.	Total $_____

Ship to:_____

Address:_____

City/ST/Zip:_____

Phone:_____

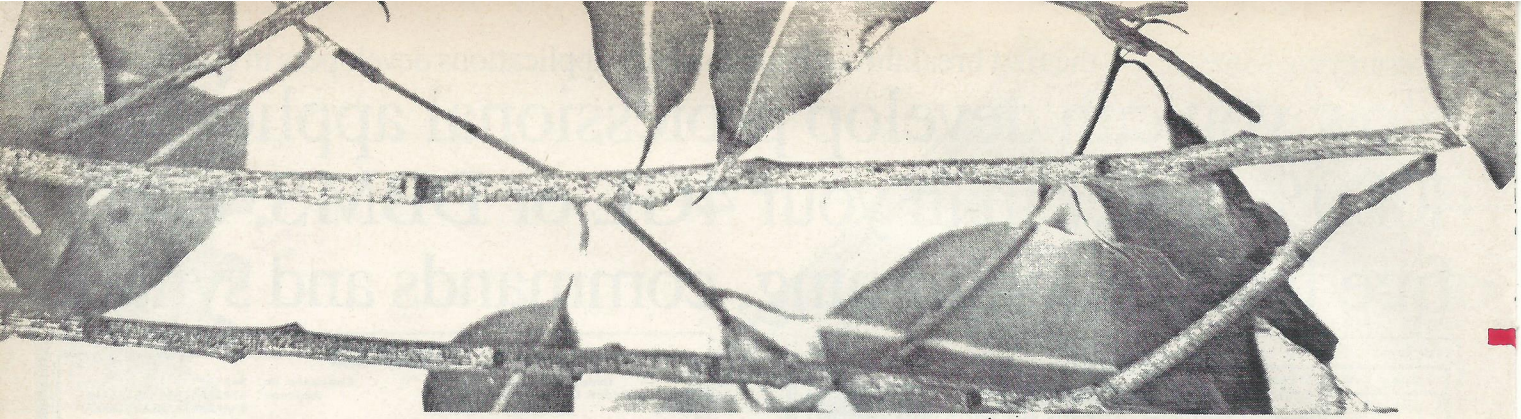
AKER

Aker Corp. 18007 Skypark Circle B2, Irvine, CA 92714
(714) 250-1718, Elec. Mail Dialcom 41:AKR 001 Telex 4931184
AKR UI OEM and VAR inquiries are welcome.

Min. requirements PC DOS 2.0, IBM PC or 100% compatible with 512K and hard disk.

©1986 Aker Corp. Printed 1/87 **Trademarks:** Magic PC, Un-Language, Window Zoom, Magic Run, Magic LAN and Magic PC Trial are trademarks of Aker Corp., IBM PC and PC-DOS are trademarks of IBM Corp., Novell is a trademark of Novell Inc.

CIRCLE 36 ON READER SERVICE CARD



This is just like our first example except that the constant for the fourth case is now 1,000 instead of 3.

It is possible to accommodate such a noncontiguous range of case values by inserting the address of *LD* in the jump table at the locations corresponding to missing case values. If only a few case values are missing, this is practical. In our previous example, however, the jump table would have to contain 1,000 addresses, 997 of which would be the address of *LD*.

To avoid this kind of space inefficiency, most compilers would compile this switch as:

```
    <code to evaluate expression e and
    put the result in register, say, r0>
    jump to LA
L0      <code for case 0>
        jump to LB
L1      <code for case 1>
        jump to LB
L2      <code for case 2>
        jump to LB
L1000   <code for case 1000>
        jump to LB
LD      <code for default case>
        jump to LB
LA      if r0=0 jump to L0
        if r0=1 jump to L1
        if r0=2 jump to L2
        if r0=1000 jump to L1000
        jump to LD
LB
```

In this situation, the jump table is replaced by a simple sequence of tests. Listings 2A and 2B give a concrete example of how this type of C switch would be translated into Motorola 68010 assembly code (UNIX-style mnemonics).

```
C program fragment--
sparse switch

switch ( eval() ) {

case 0:
   proc0();
   break;

case 1:
   proc1();
   break;

case 2:
   proc2();
   break;

case 1000:
   proc1000();
   break;

default:
   procd();
   break;
}
```

Listing 2A.

```
Equivalent M68010 code--
sparse switch

        jsr     eval
        br      L%39
L%40:
        jsr     proc0
        br      L%38
L%41:
        jsr     proc1
        br      L%38
L%42:
        jsr     proc2
        br      L%38
L%43:
        jsr     proc1000
        br      L%38
L%45:
        jsr     procd
        br      L%38
L%39:
        tst.l   %d0
        beq     L%40
        mov.l   &1,%d1
        cmp.l   %d0,%d1
        beq     L%41
        mov.l   &2,%d1
        cmp.l   %d0,%d1
        beq     L%42
        cmp.l   %d0,&1000
        beq     L%43
        br      L%45
L%38:
L%37:
```

Listing 2B.

Such a switch is approximately equivalent in terms of efficiency to a construction like:

```
temp = e;
if(temp==0){
   <code for case 0>
}
else if(temp==1){
   <code for case 1>
}
else if(temp==2){
   <code for case 2>
}
else if(temp==1000){
   <code for case 1000>
}
else{
   <code for default case>
}
```

When programming for maximum efficiency, it is important to realize this distinction: a switch whose set of constant values fills most of its range (such as {0, 1, 2, 3, 5}) is called a dense case statement and may be compiled much more efficiently than a switch with a sparse set of constant values (such as {0, 1, 2, 8761}). Of course, if the switch has only a few constant values and is not executed very often, this does not make much difference. But a nondense switch with many values is quite an inefficient construct.

If the switch is dense except for one value (for example, if the set of constant values is {0, 1, 2, ... 100, 1,000}), for maximum efficiency it may be preferable to split off the 1,000 case with an *if* statement before entering the dense switch.

As a test, I wrote a simple program that executed a switch one million times, with randomly chosen values for the case expression. The switch had 101 cases, each containing very little code. For the first run, the case constant values were the values 1 through 101. The execution time of this test program was 19 seconds on the AT&T PC7300 under the UNIX operating system.

For the second part of the test, the last case constant value was changed from 101 to 1,000. As a result, the execution

ops/83™

OUT OF THE LAB AND INTO THE WORKPLACE

OPS83 delivers expert systems power where you need it. In the lab. And in the field. Its unique features offer developers and end-users a practical, powerful and cost-effective tool for developing knowledge-based expert systems and delivering them in the real world.

OPS83 RUNS ON YOUR MACHINE

Full implementation of OPS83 is available for the:

Apollo Domain
AT&T 3B Series
HP 9000 Series 300
Sun 3
Tektronix 4400 Series
VAX, MicroVAX
IBM PC, PC/XT, PC/AT
PC Compatibles

OPS83 Unchains Your Thinking.
OPS83 is a rule-based expert systems programming language with no built-in hueristic strategy, no pre-conceived control structure. You — and you alone — choose the method that best solves your problem.

OPS83 Speaks Your Language.
Because OPS83 is written in C and conforms to C calling conventions, it can interface easily with most other languages. There's no need to discard or rewrite existing code.

OPS83 Outperforms the Competition.
When NASA performed benchmark tests on a series of software/hardware combinations, OPS83 won time after time — whether running on a supermini or a PC.

OPS83 Conserves Your Resources.
Because OPS83 compiles knowledge bases into machine code, the only part you need to link with your application program is the Runtime System. That takes up very little room — only 50 kilobytes on a PC, for example — and does not require a lot of machine resources. With OPS83 several programmers can share the same supermini or supermicro.

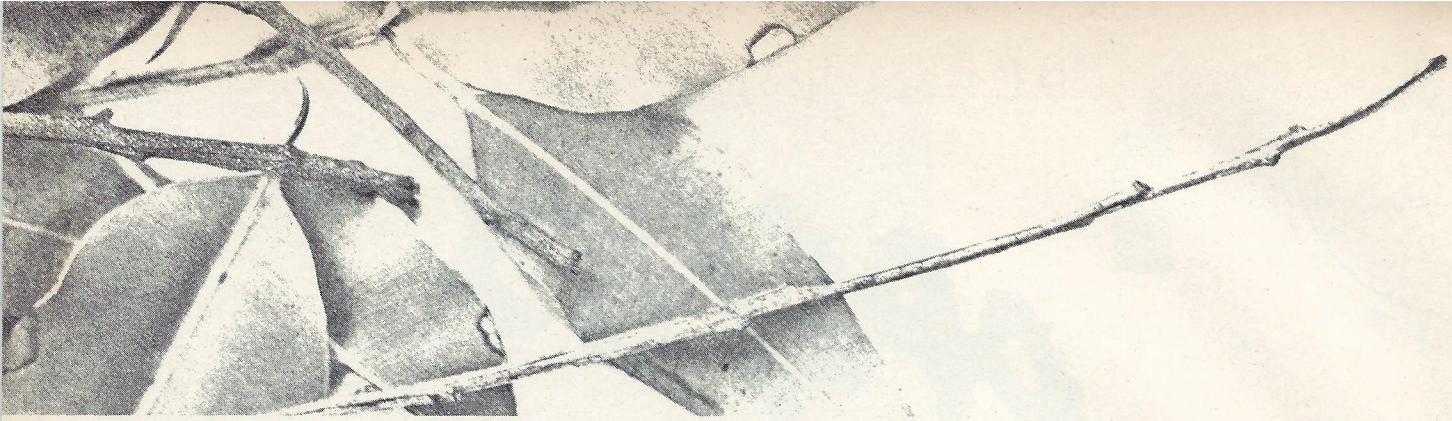
OPS83 Runs on Ordinary Hardware.
You no longer need a custom LISP machine to run high powered AI software. OPS83's proprietary compiler allows it to operate efficiently on conventional computers. Cutting lab costs and making it easy to replicate and deliver your systems on inexpensive hardware.

If you're interested in programming power in the lab and in delivering powerful, cost-effective expert systems in the workplace, write for our technical brochure and price list today. Or call us at **412-683-4000.**

Domain is a registered trademark of Apollo Computer. 3B is a trademark of AT&T Information Systems. 9000 Series 300 is a trademark of Hewlett Packard. Sun 3 is a trademark of Sun Microsystems, Inc. TEK 4400 is a trademark of Tektronix, Inc. IBM PC, PC/XT and PC/AT are trademarks of IBM Corp. VAX and MicroVAX are trademarks of Digital Equipment Corp.

PRODUCTION SYSTEMS TECHNOLOGIES INC.
5001 Baum Boulevard, Dept. D-2, Pittsburgh, Pennsylvania 15213.

CIRCLE 37 ON READER SERVICE CARD



time increased to 30 seconds. This 50% increase in run time was due to the difference between the sparse and the dense paradigms for switch compilation.

A highly optimizing compiler may use other techniques to compile switches. For example, if the set of constant values is nondense and large, the compiler may set up a table in which each entry consists of a constant value and the address of the corresponding case code, and then sort the table by constant value. Then, at execution time, the correct table entry can be found by doing a binary search of that table.

The switch control structure in C provides a natural, easy-to-read way of doing multiway branching in C and can be very efficient, particularly if the programmer is aware of how it is likely to be translated into machine instructions by a compiler.

Invoking indexed procedures

C provides another way of doing multiway branching: indexed procedure invocation. This technique is far less frequently used than the switch and may easily be abused to produce poorly structured code. When carefully used, however, it can provide an elegant solution to some multiway branching problems.

This technique is based on C's ability to compute the address of a procedure, store this address in a data structure, and, finally, call any procedure for which we have such an address.

In C, a procedure name not followed by parentheses is a constant whose value is the address of the named procedure. If we have a variable of the appropriate type, we can assign this address to it.

For example, the variable *x* is declared as *int (*x)();* so that *x* is a pointer to a function that returns an *int* and the C library function *printf* is defined in the usual way as *int printf();*. If we execute the assignment *x=printf;*, then the procedure call *(*x)("hello, world\n");* has an effect equivalent to *printf("hello, world\n");*.

By putting the addresses of several functions into an array, we can select a procedure for execution based on the value of an array index.

Suppose we have defined three C functions, *fun0()*, *fun1()*, and *fun2()*, each of which takes a single integer argument and returns an integer value. We can put the addresses of these three functions into an array, *farray[]*, with the initializing definition:

```
int (*farray[])() = {fun0, fun1, fun2};
```

Suppose we have an integer-valued expression *exp* and want to execute *fun0(arg)* if *exp* is equal to 0, *fun1(arg)* if *exp* is equal to 1, or *fun2(arg)* if *exp* is equal to 2. In every case, we want to return the value from the function into the variable *val*. Then the statement:

```
val=(*farray[exp])(arg);
```

does exactly that. This kind of vectored execution is very efficient and gives the programmer great control, but it is less familiar to many programmers. Listings 3A and 3B give a concrete example of the kind of code generated for this technique by one compiler.

Other techniques?

It is possible to imagine a third kind of multiway transfer: an indexed array of labels, as in ALGOL and PL/I.

Mercifully, the designers of C have not provided programmers with this kind of weapon. The switch control structure, with its forward-only branching, and the indexed array invocation, in which flow control always resumes with the following statement, provide run-time efficiency and readability in a way that should satisfy structured programming purists.

The expert C programmer not only is familiar with these techniques but is aware of how they are implemented by typical compilers.

Alan Filipski has a Ph.D. in computer science from Michigan State University, East Lansing, Mich. He has served on the faculties of Central Michigan University and Arizona State University and was a project leader on the first AT&T certified port of UNIX System V to an M68000 microprocessor. He is currently a member of the technical staff of GTX Corp. in Phoenix, Ariz.

```
C program fragment--indexed procedure invocation

extern int fun0(), fun1(), fun2();
int (*farray[])() = {fun0, fun1, fun2};
int exp;
test()
{
   (*farray[exp])();
}
```

Listing 3A.

```
Equivalent M68010 code--indexed
procedure invocation

farray:
        long    fun0
        long    fun1
        long    fun2

test:   mov.l   exp,%d0
        lsl.l   &2,%d0
        mov.l   &farray,%a1
        mov.l   0(%a1,%d0.l),%a0
        jsr     (%a0)
```

Listing 3B.

Artwork: Dwight Been

EVEN MORE POWER AND FLEXIBILITY

BRIEF 2.0

Users and industry press alike have unanimously proclaimed BRIEF as the best program editor available today. Now, the best gets better, with the release of BRIEF 2.0.

Straight from the box, BRIEF offers an exceptional range of features. Many users find that BRIEF is the only editor they'll ever need, with features like real, multi-level Undo, flexible windowing and unlimited file size. But BRIEF has tremendous hidden power in its exclusive macro language. With it, you can turn BRIEF into your own custom editor containing the commands and features you desire. It's fast and easy.

Jerry Pournelle, columnist for BYTE magazine summed it all up by saying BRIEF is, "**Recommended. If you need a general purpose PC programming editor, look no further.**" His point of view has been affirmed by rave reviews in C JOURNAL, COMPUTER LANGUAGE, DR. DOBB'S JOURNAL, DATA BASED ADVISOR, INFOWORLD AND PC MAGAZINE.

One user stated "BRIEF is one of the few pieces of software that I would dare call a masterpiece." Order BRIEF now and find out why. **BRIEF 2.0 is just $195. If you already own BRIEF, call for upgrade information.**

TO ORDER CALL: 1-800-821-2492 (in MA call 617-659-1571)

As always, BRIEF comes with a 30 day money-back satisfaction guarantee.

Solution Systems™
335 Washington St.
Norwell, MA 02061
(617) 659-1571

Look at these BRIEF 2.0 enhancements!

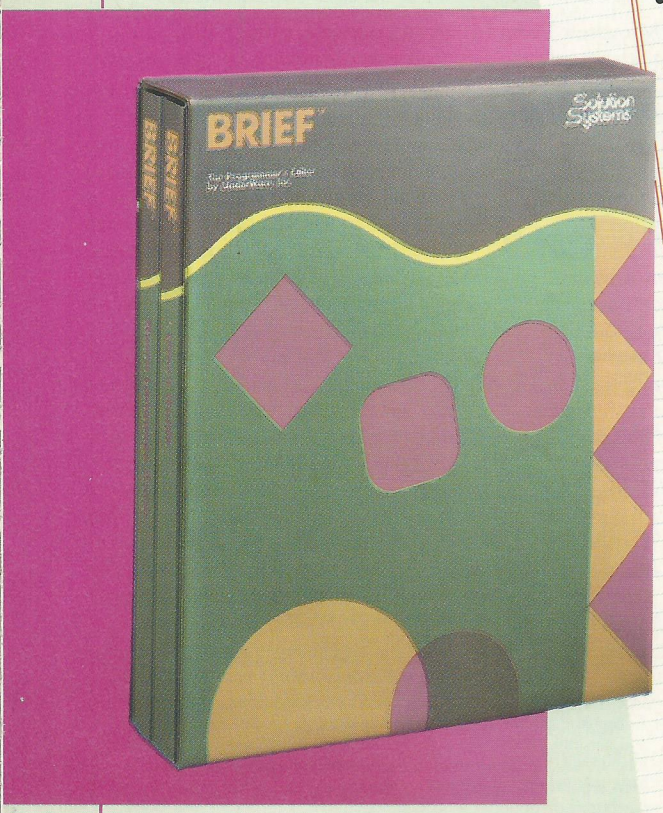
Main Features:

- All new documentation with tutorials on basic editing, regular expressions **and** the BRIEF Macro Language.
- Setup program for easy installation and configuration. (Requires no knowledge of the macro language)
- Increased speed for sophisticated operations like Undo and Regular Expression Search.
- Expanded regular expressions, with matching over line boundaries.
- More block types, with marking by character, line or column.
- Command line editing (move cursor, add and delete characters, specify command parameters).
- Support for more programming languages.
- Optional borderless windows.
- Enhanced large display support, including wider displays.
- Reconfigurable indenting for C files (supports most indenting styles).

Plus the basic features that made BRIEF SO popular!

Basic Features:

- Full multi-level Undo
- Windows
- Edit many files at once
- File size limited only by disk space
- Automatic language sensitive indentation



Requires an IBM PC or compatible with at least 192K RAM.

BRIEF is a trademark of UnderWare, Inc.

Solution Systems is a trademark of Solution Systems.

CIRCLE 53 ON READER SERVICE CARD

The fastest C

Your search for execution speed is over. The new Microsoft® C Compiler Version 4.0 is here. With blazing performance. We've added common sub-expression elimination to our optimizer that produces code that rips through the benchmarks faster than ever before.

"...the Microsoft performance in the benchmarks for program execution is the best of the lot overall." —William Hunt, *PC Tech Journal*, January, 1986.*

But speed isn't the only edge you get with Microsoft C. Other advantages include a variety of memory models like our new HUGE model that breaks the 64K limit on single data items. Plus our NEAR, FAR and HUGE pointers, which provide you greater flexibility. All this allows you to fine tune your program to be as small and fast as possible.

"Excellent execution times, the fastest register sieve, and the best documentation in this review ...Microsoft Corporation has produced a tremendously useful compiler." —Christopher Skelly, *Computer Language*, February, 1986.

No more debugging hassles. Introducing CodeView. Free.

Now, for a limited time, we'll give you an unprecedented programming tool when you buy Microsoft C, free. New Microsoft CodeView™ offers the most powerful tool yet in the war on C bugs. Forget the hex dumps. Now you can view and work with programs at any level you want. Use the program source, the disassembled object code, or

Microsoft C Compiler Version 4.00

Microsoft C Compiler

- Produces fast executables and optimized code including elimination of common sub-expressions. NEW!
- Implements register variables.
- Small, Medium and Large Memory model libraries.
- Compact and HUGE memory model libraries. NEW!
- Can mix models with NEAR, FAR and the new HUGE pointers.
- Transport source and object code between MS-DOS® and XENIX® operating systems.
- Library routines implement most of UNIX™ System V C library.
- Start-up source code to help create ROMable code. NEW!
- Full proposed ANSI C library support (except clock). NEW!
- Large number of third party support libraries available.
- Choose from three math libraries and generate in-line 8087/80287 instructions or floating point calls:
  - floating point emulator (utilizes 8087/80287 if installed).
  - 8087/80287 coprocessor support.
  - alternate math package – extra speed without an 8087/80287.
- Link your C routines with Microsoft FORTRAN (version 3.3 or higher), Microsoft Pascal (version 3.3 or higher) or Microsoft Macro Assembler.
- Microsoft Windows support and MS-DOS 3.1 networking support.
- Supports MS-DOS pathnames and input/output redirection.

Microsoft Program Maintenance Utility. NEW!

- Rebuilds your applications after your source files have changed.
- Supports macro definitions and inference rules.

Other Utilities

- Library Manager.
- Object Code Linker.
- EXE File Compression Utility.
- EXE File Header Utility.

C Benchmarks

In seconds

	Microsoft C 4.0	Lattice C 3.0	Computer Innovation C 2.3	Aztec C86 3.2	Wizard C 3.0
Sieve of Eratosthenes (register)	82.9	151.4	172.3	88.0	91.9
Copy Block	86.9	231.7	199.0	123.8	189.5

Run on an IBM PC XT with 512K memory

Microsoft CodeView
Window-oriented source-level debugger. NEW!

- Watch the values of your local and global variables and expressions as you debug.
- Set conditional breakpoints on variables, expressions or memory; trace and single step.
- Watch CPU registers and flags as you execute.
- Effectively uses up to four windows.
- Debug using your original source code, the resulting disassembly or both intermingled.
- Use drop-down menus to execute CodeView commands.
- Access the on-line help to lead you through CodeView's options and settings.
- Easily debug graphics-oriented programs since program output is kept separate from debugger output.
- Keyboard or optional mouse support.
- Enter in familiar SYMDEB or DEBUG commands.

*Reprinted from *PC Tech Journal*, January 1986, copyright 1986, Ziff-Davis Publishing.

you've ever seen.

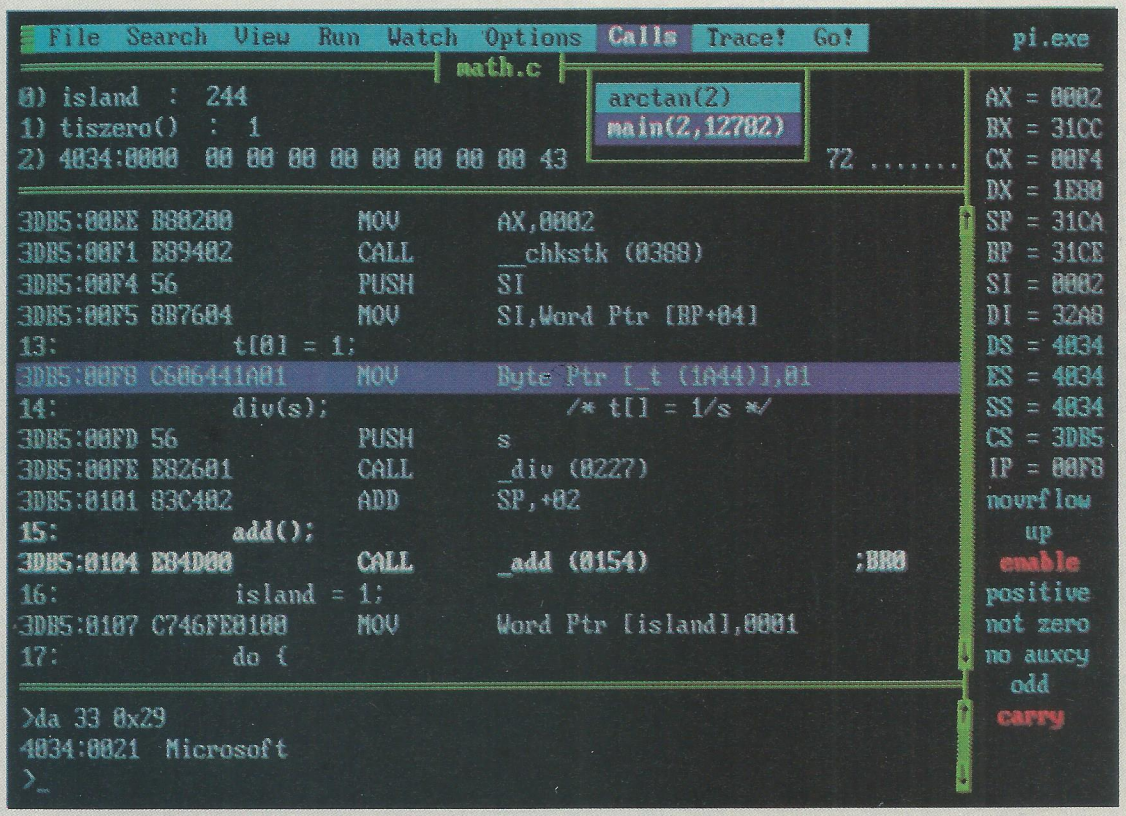
both at the same time. Open a window to view CPU registers and flags. Watch local and global variables as well. All while your program is running.

CodeView gives you complete control. Trace execution a line at a time—using source or assembly code. Or set conditional breakpoints on variables, memory or expressions. CodeView supports the familiar SYMDEB command syntax, as you'd expect. Commands are also available through drop-down menus. Combine the new window-oriented interface with our on-line help and debugging has never been easier. Or quicker.

Take the $5 CodeView tour.

You may find it hard to believe our debugger can do all we've claimed. So we're offering test drives. Five bucks will put you behind the wheel of a Microsoft C demo disk with CodeView.† See for yourself how fast debugging can get.

For more information about the CodeView demo disk, the new Microsoft C Compiler, a list of third party library support or the name of your nearest Microsoft dealer, call (800) 426-9400. In Washington State and Alaska, (206) 882-8088. In Canada call (416) 673-7638.



Microsoft® C Compiler

The High Performance Software

Microsoft, MS-DOS and XENIX are registered trademarks and CodeView is a trademark of Microsoft Corporation. UNIX is a trademark of AT&T Bell Laboratories. IBM is a registered trademark of International Business Machines Corporation. †Offer expires 12/31/86.

CIRCLE 39 ON READER SERVICE CARD

PAINLESS WINDOWS.

Windows. Data Entry. Menus. Finally, a C programmers' tool that makes them as easy to use as *printf()*. With Greenleaf DataWindows™, you move in quantum leaps!

Snazzy Window Treatments

DataWindows represents an important breakthrough in C programming tools. It sets you free so you can create exciting programs quickly and easily, saving both time and money! Developed to work with the IBM PC, XT, AT, compatibles, and MSDOS or PCDOS, DataWindows is a carefully tooled system of C functions which will jazz up your programs with unprecedented efficiency.

Greenleaf DataWindows is integrated windows, transaction data entry, pop-up, pull-down, and Lotus style menu systems with:

- **Screen Management.** You don't have to remember what's on the display or the sequence in which you put it there. DataWindows does the grunt work. There are no restrictions.
- **Transaction Data Entry.** Data entry windows can have any number of fields with sophisticated options for reading many data types. Calls are made to help, validation, and other functions. Full featured text editing, protected and mandatory fields, dBASE type picture strings, context sensitive help, validation of fields and transactions, redefinable keys, password entry, attribute control, keyboard idle and much more.
- **Device Independence.** It detects the type of display adapter your computer is using and adjusts to it automatically for CGA, EGA, or monochrome. Logical video attributes are easy to use for color or monochrome.
- **Compatibility.** Runs with Microsoft Windows and IBM TopView.
- **The Greenleaf Tradition of Quality.** Reliable products. Professional documentation that gets you up and running quickly and keeps you there. Reference card. Newsletter and Bulletin board.

IBM, Microsoft & dBase, are registered trademarks of International Business Machines, Microsoft Corporation & Ashton-Tate respectively.
PCDOS, IBM PC, XT, AT, & TopView are trademarks of IBM; MSDOS and Microsoft Windows are trademarks of Microsoft Corporation.

Stop Window Shopping

Order Today. Or call toll free for a free demo of the windows library that makes all the others obsolete.

Order any of these high performance tools by calling your dealer or 1-800-523-9830 today. Specify compiler when ordering. Add $8 for UPS second day air, or $5 for ground. Texas residents add sales tax. MasterCard, VISA, P.O., check, COD. In stock, shipped next day.

Greenleaf DataWindows	$225
DataWindows Source Module	$225
The Greenleaf Comm Library v2.0	$185
The Greenleaf Functions v3.0	$185
Digiboard Comm/4-II	$325
Digiboard Comm/8-II	$535

GREENLEAF Software©

1411 LeMay Drive, Suite 101
Carrollton, TX 75007

Call Toll Free
1-800-523-9830
In Texas and Alaska, call
214-446-8641

CIRCLE 40 ON READER SERVICE CARD

Window Dressings

- **Simple or Complex Windows.** Up to 254 powerful overlaid windows simultaneously, all with just one kind of window to remember! Yet any window can be from one character to 32K!
- **Easy Window Operations.** DataWindows lets you move, zoom, frame, title, change colors, titles, frames, size, location, and make windows visible or invisible at will! Functions set cursor, attributes, and write data to any window or "current window". Word wrap, auto scroll, keyboard functions.
- **Write to Any Window Any Time.** Windows may be visible, overlaid, or invisible, and you can write to them anyway. What you write will be seen when the windows become visible.
- **DataWindows is fast!** It writes directly to video memory (in some modes).
- **Easy to save!** Any window, complete with attributes, can be saved on disk quickly and efficiently.
- **Source code available. No royalties.**

Also from Greenleaf:

The Greenleaf Functions v3.0
The most complete, mature C language function library for the IBM PC, XT, AT and close compatibles. Includes over 225 functions — DOS, disk, video, color text and graphics, string, time/date, keyboard, disk status and Ctrl-Break functions plus many more.

The Greenleaf Comm Library
Our 2.0 version is the hottest communications facility of its kind. Over 120 functions — ring buffered, interrupt driven asynchronous communications for up to 16 ports simultaneously with XMODEM, XON/XOFF, many many sophisticated features.

We support all popular C compilers for MSDOS/PCDOS: Microsoft, Lattice, Computer Innovations, Aztec, DeSmet, and others.

PUBLIC DOMAIN SOFTWARE REVIEW

Software metrics and EDDY

By Tim Parker

One of the best things about being a public domain columnist is that occasionally a package containing a new program arrives in the mail from a reader or business firm.

Most of the material I've received lately has been of the shareware variety, of which there are two conflicting points of view. The first is that authors are asking for donations while previously they would have simply let it go for free.

The second is that a growing number of products that would have been marketed commercially are now listed as shareware, with the "Try it and if you like it, pay us" philosophy. In these days of expensive software, the latter is definitely a boon to the consumer. The first point of view cannot really be disputed because most authors will spend the extra time to create a package of commercial quality if they can expect some reward for their effort.

Indeed, the last point has been borne out by discussion with many programmers. Take a hypothetical example: I need a routine to do syntax checking on my WHYBOL-47 programming language. Normally I'd hack the thing together with a simple interface, and no menus or help screen and use it as a standard tool in my WHYBOL programmer's toolkit. But I would be tempted to clean up the package and let it go shareware with a donation request if I knew that other programmers were interested in WHYBOL-47, and the only commercially available syntax checker, offered by Fly-By-Night Programming Inc. for $600, wasn't much better than mine.

Even if I request only a $25 donation and 10 people send in the money, the time to write the interface has paid for itself, and my self-esteem and reputation with other programmers have increased.

Now, let's approach the discussion from the other side: as a harried programmer tackling WHYBOL-47 for the first time, and performing under deadline. Instead of paying the $600 to Fly-By-Night, I would get the shareware product from a bulletin board. If it cost me $25, what the heck; I'd still save a fortune, and I wouldn't have to write the code myself.

This example is a little silly and the circumstances are contrived. But the number of shareware users is increasing at a phenomenal rate, and the number of excellent shareware packages is increasing steadily.

Unfortunately, the percentage of users who actually pay for their shareware is small. Two or three thousand people may pick up a typical shareware package off bulletin boards. Normally, fewer than 100 of these people will pay the donation or register their copies. Programmers are not getting rich because of shareware, but they are being compensated somewhat for their time.

I received a particularly interesting disk a short time ago containing two products that made me think of the example just mentioned: METRIC and PCDRAFT, from SET Laboratories Inc., P.O. Box 03963, Portland, Ore. 97203, (503) 289-4758.

One of the packages, METRIC, is very interesting, and I don't think I've seen a similar commercial or public domain release that does quite the same task: it is designed to determine the complexity of a Pascal program.

Software metrics is a relatively new field that is based on a practice programmers have been performing from the start. Software development, especially in a corporate environment, requires a considerable amount of structuring, reporting, estimating, and costing. Software usually has to be cost effective to justify its development and testing time. In addition, modifications are usually required, and it is important that someone who has never seen it before be able to understand the program code. The amount of time and programming effort needed to code a program has always been assumed to be directly related to the complexity of the code itself.

Complexity is measurable in a number of ways. The easiest way is to count the number of decision statements. Although this technique is crude, the more decision statements, the longer (and hence more complex) the code is. This is a poor first approximation, however, as a little thought will show. If a program is constructed modularly, modifications may only be required to a single module while the programmer ignores the rest of the routines.

Further, any code segments that are executed only under certain conditions and are not required for the modifications are ignored. If simple decision structures are used (such as $X=Y$) instead of complicated structures with numerous arguments, the code is easier to read and modify.

The sheer size of a program is easy to determine. If only nonblank lines are considered and all comments are ignored, the number of lines in a program will give a first impression of its complexity. Next, you can get a more accurate idea of a program's complexity by counting the number of tokens in the code. (A 1,000-line program with an average of three words per line is less complex than a 1,000-line program with 20 words per line.)

The number of branching statements will give a first approximation of the complexity of the program control flow. A program with numerous *IF*s, *WHILE*s, and similar structures is more complex than a top-to-bottom program with no branching or flow interruptions.

Altogether, these procedures develop into an assessment of the complexity of the program expressed by a number of variables. By taking these variables into consideration, a program's development time can be estimated. Indeed, literature has shown that these assessments are fairly accurate when a good complexity rating algorithm is used—much more accurate than other estimating routines that are used by development people.

So far the discussion has dealt with determining the complexity of existing programs. If these existing programs can be assessed, it would seem logical that a currently undeveloped program can be placed on the scale of complexity, based

on a person's experience and instinct, to derive a better estimate of the development time than would be possible through simple guesswork.

For those who are interested in software metrics, a short bibliography of references is contained in the accompanying sidebar, "Suggested reading on software metrics."

METRIC, when run against some Pascal source code, uses algorithms to determine the complexity of the code. A discussion of the algorithms and techniques is included with the METRIC program in the documentation file, along with a brief description of the software metric field.

The interface is simple. After a single identification screen, the input file name is requested. The screen then shows the ascending word count and the procedure currently being evaluated. After completion, the program exits and a text file that lists the METRIC output is created.

The output is simple. It lists a number of software metric variable results, which are correlated to the variable's meaning (reference books are best here). A number is produced for the code's volume and effort. A predicted bugs number shows an estimate of the number of errors that can be generated (this is not a syntax check!), and a predicted development time is shown in minutes and hours. A cyclomatic number is also produced, which again requires the user to know a little about software metric theory. The number of lines of code, procedures, and executable semicolons is printed.

The program allows the user to modify the list of words recognized as tokens by METRIC, enabling the system to be customized to different compilers. For example, a Turbo Pascal user can include all the Turbo keywords not included in the token list to get an even more accurate idea of the program complexity.

As a sample, I ran the program FICALCS through METRIC. (I discussed FICALCS several months ago in "Financial calculations and Batch files," *COMPUTER LANGUAGE*, Jan. 1987, pp. 83-85. Written by Roy MacLean, FICALCS is a great financial calculator and a good example of Turbo coding.) METRIC predicted 16 bugs in the finished code.

After I customized the METRIC package to Turbo keywords, the prediction dropped to two (none actually exist in the package: bugs are predicted based on the complexity of the code). The estimated development time was 311 hours, which, according to MacLean, is remarkably close (an actual count wasn't kept, but this seems as good an estimate as is possible). The almost 1,500 lines of code had a purity ratio of 0.51, which does not mean his code is only half pure. (Read the books for an explanation.)

On one of my network programs, designed as an on-line help system, METRIC produced a development time of 120 hours. The actual time was 127. It was remarkably close on several other test program too, although the errors increased as the hours both exceeded 200 and went below 10. Nevertheless, I was amazed at the accuracy of the estimates.

Granted, the information METRIC supplies isn't very useful to the run-of-the-mill programmer who wants to hack out a simple game, but it is to serious software developers. After using the program for a few weeks on random selections of code that are either finished or under development in my library, I got very interested in the subject and followed through the readings in the bibliography. Unconsciously, I found myself coding a little differently, in a more structured and modular way than before.

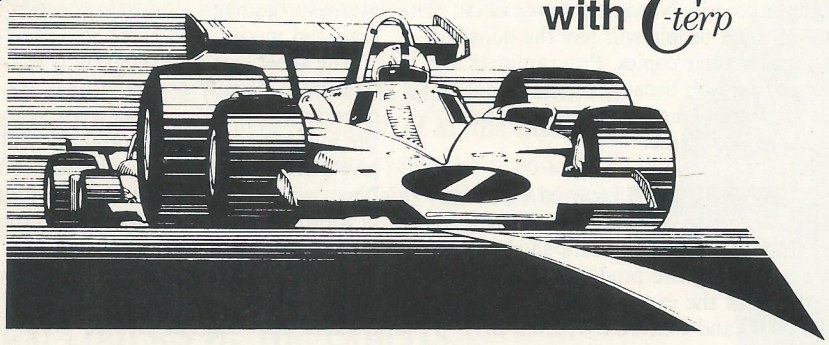
METRIC is sure to get readers who are interested in software design theory hooked on the field. I was surprised to find that many software development houses do follow the metric field quite intensely, although not always under that name or with the level of complexity that is possible.

METRIC is available on several bulletin boards and directly from SET Laboratories. The author requests a $99 donation if the software is found useful.

I suspect METRIC will have a very limited appeal, but for those who revel in

#1 C interpreter

Turbo-charge your C compiler with C-terp



Our C Interpreter provides the finest and fastest development environment for C and is compatible with your compiler.

- **Fast Semi-Compilation** -- We convert source to tokens faster than any product (existing or announced) on the market.
- **Interactive Debugging** -- See your code come to life as you single step, set breakpoints, call functions, view data, execute any C expression.
- **Complete Language** -- We've always supported full K&R, now we support the usual ANSI enhancements as well (structure assignment, enumerations, etc.) as well as keywords cdecl and far.
- **Multiple Modules** -- an accurate reproduction of a typical multiple module compiler environment brought to you in a high speed interactive interpreter.
- **Multi-file, configurable editor** -- features fast screens, inter-file copies and moves, etc. etc. Spring from file to file, module to module. Develop as you never did before. Completely reconfigurable.
- **Complete Compatibility** -- For each supported compiler we provided a separate C-terp with separate documentation (each compiler is a little bit different). We provide a batch file to link in your compiler's entire library. We make sure the data alignment, bit field order, and pre-processor variables are compatible with your compiler. We care about compatibility.
- **Shared symbols option** -- for those large 75-module applications.
- **Software Paging** -- for those big jobs. Our new and improved paging can now access Extended Memory directly.
- **Pointer checking** -- An out-of-bounds assignment will put you into debug mode with the offending statement highlighted.
- **Object module support** -- Link in not only your compiler's library but your own libraries (large model), assembler routines, and commercial libraries such as Essential Graphics, HALO, Windows for Data, Greenleaf, Vitamin C, etc. Our function pointers are compatible with compiled C (a must for using commercial libraries) and we support call in (from compiled to interpreted) as well.
- **Numerous other features** including our own batch mode, dual display and graphics support, tracing and 8087/80287 as well.

Order C-terp TODAY (Specify Compiler)
Microsoft, Lattice, Aztec, C86, Mark Williams, Xenix

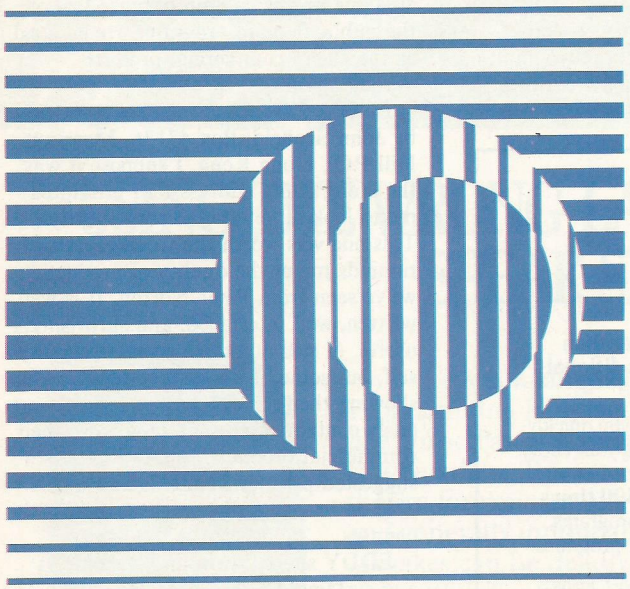
PRICE: MS-DOS 2.x and up - $298
Xenix 286 System V - $498

VISA, MC, COD * 30 Day Money Back Guarantee

* C-terp is a trademark of Gimpel Software.

GIMPEL SOFTWARE
3207 Hogarth Lane * Collegeville, PA 19426
(215) 584-4261

FINALLY...EXPERT SYSTEMS IN MAINSTREAM ENVIRONMENTS



Logicware's expert systems tools, TWAICE™ and MProlog™ let you develop and deliver expert systems on IBM mainframes, DEC VAX, M68000 workstations and IBM PC.

TWAICE Expert System Shell:
TWAICE, an expert system shell and knowledge engineering environment, enables you to create sophisticated rule-based expert systems that store factual information, reason with that information, answer questions and conduct conversations in an intelligent fashion.

MPROLOG Language:
MProlog, the language underlying TWAICE, is modular, fast and highly portable. MProlog is a general-purpose programming language, but is especially suited to developing specialized AI applications.

AI EDGE™ Program:
The AI EDGE program is a low risk introduction to expert system technology. Logicware's knowledge engineers profile an application in your chosen area and develop a prototype expert system. This six-week project will give you the experience you need to assess the prospects of expert systems in your workplace.

EDUCATION Programs:
Logicware offers complete and comprehensive education on our expert system tools and their applications.

ENVIRONMENTS:
TWAICE and MPROLOG are available on IBM MVS/TSO and VM/CMS, DEC VAX and Micro VAX, Nixdorf Targon 31-35, Pyramid 90x, SUN, ISI, Apollo, Tektronix 4404-6, HP 3000, CDC Cyber 180 and IBM PC/AT.
As well, MProlog is available on IBM PC/XT (and compatibles) and ATARI ST.

For more information, write or call:

Logicware Inc.
237 Park Avenue
Suite 2136
New York
N.Y. 10017
(212) 551-3536

Logicware Inc.
5915 Airport Rd.
Suite 200
Mississauga
Ontario, Canada
L4V 1T1
(416) 672-0300

Logicware Inc.

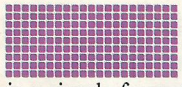
TM–IBM, MVS, VM/CMS: International Business Machines, Inc./UNIX: Bell Labs/SUN: Sun Microsystems Inc./Apollo: Apollo Computer Corp./ISI: Integrated Solutions Inc./Tektronix: Tektronix Corporation/Targon: Nixdorf Computer AG/Pyramid: Pyramid Technologies/Twaice: Nixdorf Computer AG: M68000: Motorola Inc.

R–DEC VAX: Digital Equipment Corporation.

HP 3000: Hewlett–Packard Inc./Cyber: Control Data Corporation/1040ST: Atari Corporation

CIRCLE 54 ON READER SERVICE CARD

software theory and find themselves in software development cycles, it is an interesting, useful, and innovative program.



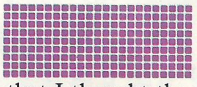
The other product sent to me by SET Laboratories is a simple formatter, similar in many respects to nroff formatters on mainframes and minis. It is worth mentioning because it is one of the few post-processor text formatters I have found that works consistently bug free. It also has a good arsenal of command functions and works fairly quickly.

PCDRAFT commands are all on new lines and begin with a period, like WordStar dot commands. A brief list of the functions supported includes page eject, centering, line spacing, page forcing, display text, footers, headers, item lists, auto indent, margin settings, page number changing and printing, page length, autoindent on paragraphs, section headers, and underlining.

Although many people prefer word processors to text formatters, when the need for a test formatter arises, I find PCDRAFT to be very useful. Its command set is complete enough to allow all the formatting normally required, and it has a handy automatic table of contents generator that is based on sections titles.

I found I could port files without modifying them from the local university Amdahl to my PC and back using PCDRAFT and a modified version of vroff on the mainframe, thereby simplifying documentation on some mainframe code I am writing. Since word processors are still scarce on mainframes and many minis, ASCII files and text processors are de rigeur in that environment. Although not true of micros, it is necessary to use these postprocessor text formatters when working in both environments, and PCDRAFT is a good one.

SET Laboratories requests a $20 donation if PCDRAFT is found useful.



I have seen so many disk directory utilities over the years, that I thought the market was saturated. I was wrong. I also thought I'd never discuss one in this column because few of the public domain programs are as good as the cheap commercial products. Wrong again!

EDDY (EDit DirectorY) was written by John Scofield, Box 47136, Morrison Hill P.O., Hong Kong. I suppose it was the Hong Kong address that prompted me to look at yet another disk utility. This one was worth the trouble. EDDY reminds me of the quality of product we've seen from Peter Norton. It is well written, well interfaced, and surprisingly useful. (I still don't use an interface myself, but some of my colleagues do. After showing them EDDY, a couple have seriously contemplated switching from their commercial products.) EDDY carries a suggested price of $15. It is also available on several bulletin boards across the country.

EDDY should work on all monitors and systems. I tested it on Hercules, CGA, MDA, and EGA. A command line parameter optimizes monochrome operation.

The interface, as mentioned, is elegant. Reverse video is used where necessary, and a very good on-line help file is available with the F1 key.

One very nice feature EDDY has that others don't is *UNDO*. Suppose you've just renamed files, changing their attributes, and then realized you'd made a mistake. One keystroke undoes it all. Date, time, file name, file type, and attributes can all be edited. The interface uses the "Put the cursor where it has to go and type" approach, allowing fast entry. (It worked with my mouse.) I found the interface particularly useful for changing file attributes.

All the major commands are available on function keys listed at the bottom of the screen. A summary of the commands not listed on the bottom line is available through the help screens.

The *delete* function works as expected, but there is a similar function called *destroy*. This function not only deletes but overwrites the data on the occupied clusters, preventing recovery with one of the superutilities like Mace or Norton.

Copy, move, and *edit* work as would be expected from a disk utility program,

Complete C Programs in Half the Time, with *Instant-C*™

***Instant-C* helps you create a working, well-tested program *faster* than any other interactive C development system. Much faster than traditional compilers, linkers, and debuggers.**

Hard to believe? Here's how we do it. Because *Instant-C* is a high-performance interpreter there are **no compile or link delays.** Change your program, then test it immediately. No matter how large your program, the turnaround time is just seconds.

"Instant-C means instant gratification."—*PC Magazine,* **Editor's Choice** for best C interpreter. *10/29/85*

"Time after time, the Instant-C prompt was starting back just barely after pressing Enter."—*PC Tech Journal, 5/86*

Source-level debugging saves your time.

- set any number of conditional breakpoints in your program;
- stop execution from keyboard;
- single-step by source statement;
- examine and change variables or code, and continue execution;
- execute any statement or function directly for instant testing;
- display source code back-traces;
- source code animation;
- monitor data changes;
- full-screen and multi-screen support, even with non-standard graphics devices.

"The resulting debugging and testing capabilities are fantastic and the detailed trace/debug/display commands make it easy."—*The C Journal, 5/85*

Run-time checking stops your program as soon as errors occur, when bugs are easiest to understand and fix.

- pointer references checked for reasonableness;
- array indexes checked within declared bounds.

Not only does *Instant-C* help you quickly change, test, check and debug your code, but it runs your program faster than any other C interpreter. **Fifty to 500 times faster!** Fast enough for *real* programs, even fast enough for real-time programs.

"It is much faster than any of the other products mentioned and was the only one able to complete the standard SIEVE in a reasonable time. Clearly, this high speed allows much more complex problems to be attacked with Instant-C than with any of the other products discussed."—*Computer Language, 2/86*

Immediate feedback and more than 400 diagnostics makes *Instant-C* great for learning C. Full K&R and the ability to **link compiled object code and libraries** (Lattice and Microsoft) makes *Instant-C* compatible with your existing programs.

"When you get right down to it, I don't think there's a better way you could learn C."—*Programmer's Journal, 3/85*

"Clearly, Instant-C is the performance champion."—*PC Tech Journal, 5/86*

The bottom line for your business is increased productivity. The result for you is a job well done, and quickly.

"We sincerely feel that Instant-C can have a major positive impact on programmer productivity."—*Computer Language, 2/85*

"Instant-C by Rational Systems is a C programmer's dream."—*Micro/Systems Journal, 3/86*

Version 2 is available for MS-DOS and PC-DOS, and comes with a full 31 day **money back guarantee.** *Instant-C* is only $495. Order today! Call or write for full information.

Rational Systems, Inc.
P.O. Box 480
Natick, MA 01760
(617) 653-6194

CIRCLE 42 ON READER SERVICE CARD

so I will not discuss them. The program can check to see that programs moving across directories do not conflict with existing file names. Also available are several command line options that control copying and date and time stamping, depending on the user's requirements.

A patch command changes a file's contents, in either hex or ASCII. Conversion between the two formats is fast. Changes are not written to the file until an *UPDATE* command is executed, to prevent an error from occurring during the patch process. The *UNDO* feature is available to get rid of bad changes.

EDDY comes with a 37-page documentation file that describes the program and its capabilities completely. Overall, I found EDDY to be a clean, easy-to-use, well-written program. John Scofield has done a good job here. ■

Tim Parker is a systems programmer in Ontario, Canada who also writes for several technical magazines.

Suggested reading on software metrics

Elements of Software Science. M. Halstead, New York: Elsevier North Holland, 1977.

"Applying Software Metrics to Program Maintenance." *IEEE Computer.* W. Harrison, et al. (Sept. 1982):65-79.

"A Complexity Measure." *IEEE Transactions on Software Engineering.* T. McCabe (Dec. 1976):308-320.

"Defining Software Science Counting Strategies." *ACM SIGPLAN Notices.* N. Salt (March 1982):58-67.

The Advanced Programmer's Editor That Doesn't Waste Your Time

EPSILON

- Fast, EMACS-style commands—completely reconfigurable
- Run other programs without stopping Epsilon—concurrently!
- C Language support—fix errors while your compiler runs
- Powerful extension language
- Great on-line help system
- Multiple windows, files
- Regular Expression search
- Unlimited file size, line length
- Supports large displays
- 30 day money-back guarantee
- Not copy protected

Only $195

Lugaru Software Ltd.

5740 Darlington Road
Pittsburgh, PA 15217

Call (412) 421-5911

for IBM PC/XT/AT's or compatibles

CIRCLE 59 ON READER SERVICE CARD

UNIX/C WINDOW DEVELOPMENT COMPATIBILITY with CURSES for MS-DOS. CALL NOW FOR MS OS/2

Buy the window development tool that Unix/C experts demand!

Let the speed, versatility and ease of using Aspen Scientific's Curses for MS-DOS improve your products' user interface—all this plus Unix/Xenix source code compatibility for you, the serious developer.

Our Curses can do any job you require from a window tool. It will save you time, money, and your reputation. Our Curses blazes on IBM PC's and compatibles, but is versatile enough to work under MS-Windows and on any personal computer running MS-DOS 2.0 and later.

Look at a few of the applications our Curses is benefiting:

- Voice Recognition
- Telephone Switching
- Environmental
- Aerospace
- Financial
- Consulting
- Robotics
- Engineering
- Hobbyist

FREE - FAST form development tool box with source code. Unix compatible. Included if you *ORDER RIGHT NOW*

Compilers: Microsoft, Lattice, Aztec, DeSmet
Trademarks: MS-DOS, MS-Windows (Microsoft); Unix (AT&T Bell Labs); Lattice (Lattice, Inc.); Aztec (Manx Software Systems); DeSmet (DeSmet Software).

Complete curses tool kit: $119.
Source code available for: $289.

ASPEN SCIENTIFIC

P.O. BOX 72 WHEAT RIDGE, COLORADO 80034-0072
(303) 423-8088

CIRCLE 88 ON READER SERVICE CARD

THE PROGRAMMER'S SHOP

helps save time, money and cut frustrations. Compare, evaluate, and find products.

RECENT DISCOVERY

Periscope III - debugger with 64K protected RAM and breakout switch; breakpoints for hardware, memory, port, data. Real-time trace buffer, pass counter. PC $ 829

AI-Expert System Dev't

Arity Combination Package	PC	$ 979
System - use with C	MS	$ 229
SQL Dev't Package	MS	$ 229
Auto-Intelligence	PC	$ 739
Exsys	PC	$ 309
Runtime System	PC	$ 469
Insight 1	MS	$ 75
Insight 2+	MS	$ 379
Intelligence/Compiler	PC	$ 739
T.I.: PC Easy	PC	$ 435
Personal Consultant Plus	PC	$2589
Personal Consultant Runtime	PC	$ 85
Turbo Expert-Startup(400 rules)	PC	$ 129
Corporate (4000 rules)	PC	$ 359

AI-Lisp

Microsoft MuLisp 85	MS	$ 159
PC Scheme LISP - by TI	PC	$ 85
Star Sapphire	MS	$ 459
TransLISP - learn fast	MS	$ 89
TransLISP PLUS		
Optional Unlimited Runtime		$ 139
PLUS for MSDOS		$ 179

Others: IQ LISP ($155), IQC LISP ($269)

AI Prolog

APT - Active Prolog Tutor - build applications interactively	PC	$ 49
ARITY Prolog - Interpreter	PC	$ 229
COMPILER/Interpreter-EXE	PC	$ 569
Standard Prolog	MS	$ 77
MicroProlog - Prof. Entry Lev.	MS	$ 85
MicroProlog Prof. Comp./Inter.	MS	$ 439
MPROLOG P550	PC	$ 175
Prolog-86 - Learn Fast	MS	$ 89
Prolog-86 Plus - Develop	MS	$ 229
TURBO PROLOG by Borland	PC	$ 69
Turbo Prolog Toolbox	PC	$ 69

Basic

BAS_C - economy	MS	$ 179
BAS__PAS - economy	MS	$ 135
Basic Development System	PC	$ 105
Basic Development Tools	PC	$ 89
Basic Windows by Syscom	PC	$ 95
BetterBASIC	PC	$ 129
Exim Toolkit - full	PC	$ 39
Finally - by Komputerwerks	PC	$ 85
Inside Track	PC	$ 50
Mach 2 by MicroHelp	PC	$ 55
Peeks n Pokes	PC	$ 35

FEATURES

COBOL I/Q - Ad hoc queries, customizable reports, and graphs from MicroFocus, RM, Microsoft, mbp COBOL, Lotus, dBASE, Btrieve, ASCII files. PC $ 539

NET-TOOLS - Access NETBIOS-compatible network systems from Microsoft C, Pascal, FORTRAN, Assembler, Lattice C. Full Source, No Royalties. PC $ 129

FREE Newsletter

Insightful commentary, guest columnists, survey results, and valuable resource listings. Interviews, technical articles, predictions — even cartoons. No wonder 96% of our readers pass *The Programmer's Letter* on to their friends; no wonder 72% make sure they get their copy back to keep for reference! You can request a FREE sample copy today by calling our toll-free number. A personal subscription is just $25 per year.

Our Services:

- Programmer's Referral List
- Compare Products
- Help find a Publisher
- Evaluation Literature FREE
- BBS - 7 PM to 7 AM 617-740-2611
- Dealers Inquire
- Newsletter
- Rush Order
- Over 700 products
- National Accounts Center

Basic Cont.

QuickBASIC	PC	$ **69**
Stay-Res	PC	$ 75
Turbo BASIC - by Borland	PC	$ 69

Cobol

Microfocus Professional Cobol	PC	$1699
VS Workbench	PC	$3379
Microsoft COBOL	MS	$ 439
Microsoft Cobol Tools	PC	$ 209
Personal COBOL	PC	$ 129
Realia - very fast	MS	$ 819
Ryan McFarland COBOL	MS	Call
COBOL-8X	MS	Call
Screenplay - screen mgmt.	PC	$ 129

C Libraries-Communications

Asynch by Blaise	PC	$ 125
Essential Comm Library	PC	$ 125
With Debugger	PC	$ 189
Greenleaf Comm Library	PC	$ 129
Multi-Comm - add multitasking	PC	$ 135

dBASE Language

Clipper compiler	PC	Call
dBASE II	MS	$ 329
dBase III Plus	PC	$ 429
dBASE III LanPack	PC	$ 649
DBXL Interpreter	PC	$ 139
FoxBASE+ - single user	MS	$ 349
QuickSilver by Word Tech	PC	$ 499

dBASE Support

dBase Tools for C	PC	$ 65
dBrief with Brief	PC	Call
dBC ISAM by Lattice	MS	Call
dBx Translator to C	MS	$ 319
dFlow - flowchart, xref	MS	Call
Documentor - dFlow superset	MS	Call
Genifer by Bytel-code generator	MS	$ 299
QuickCode III Plus	MS	$ 239

Editors for Programming

BRIEF Programmer's Editor	PC	Call
EMACS by UniPress Source: $895		$ 265
Epsilon - like EMACS	PC	$ 149
KEDIT - like XEDIT	PC	$ 99
Micro Focus Micro/SPF	PC	$ 139
PC/EDT - macros	PC	$ 229
PC/VI - by Custom Software	MS	$ 109
Personal REXX	PC	$ 99
PMATE - power, multitask	PC	$ 119
SPF/PC - fast, virtual memory	PC	$ 189
Vedit	MS	$ 99
Vedit PLUS	MS	$ 129

RECENT DISCOVERY

CxPERT - Expert systems shell, translates to C code to integrate with your application. Certainty factors, explanations, inheritance, frames, help. MS $295

C Language-Compilers

AZTEC C86 - Commercial	PC	$499
C86 PLUS - by CI	MS	$379
Datalight C - fast compile, good code, 4 models, Lattice compatible, Lib source. Dev'rs Kit	PC	$ 77
Datalight Optimum-C	MS	$109
with Light Tools by Blaise	PC	$168
Lattice C - from Lattice	MS	$269
Let's C Combo Pack	PC	$ 99
Let's C	PC	$ 57
Microsoft C 4.0- Codeview	MS	**$275**
Rex - C/86 - standalone ROM	MS	$695
Turbo C by Borland	PC	$ 69
Uniware 68000/10/20 Cross Compiler by SDS	MS	Call
Wizard C	MS	$299
ROM Development Package	MS	$259

C Language-Interpreters

C-terp by Gimpel - full K & R	MS	$219
C Trainer - by Catalytix	PC	$ 89
INSTANT C - Source debug, Edit to Run-3 seconds, .OBJs	MS	$369
Interactive C by IMPACC Assoc.	PC	$209
Run/C Professional	MS	$155
Run/C Lite	MS	$ 79

C Libraries-General

Blackstar C Function Library	PC	$ 79
C Essentials - 200 functions	PC	$ 75
C Function Library	MS	$109
C Tools Plus (1 & 2) - Blaise	PC	$125
C Utilities by Essential	PC	$129
C Worthy Library - Complete, machine independent	MS	$249
Entelekon C Function Library	PC	$119
Entelekon Superfonts for C	PC	$ 45
Greenleaf Functions-portable, ASM		$139
LIGHT TOOLS by Blaise	PC	$ 69

C Libraries-Files

C Index by Trio - full B+ Tree.	MS	$ 89
/File is object only	MS	$ 89
/Plus is full source	MS	$319
BTree by Soft Focus	MS	$ 69
CBTREE - Source, no royalties	MS	$ 99
CTree by Faircom - no royalties	MS	$315
rtree - report generation	PC	$239
dbQUERY - ad Loc, SQL - based	MS	$129
dbVISTA - full indexing, plus optional record types, pointers, Network.		
Object only - MS C, LAT, C86		$129
Source - Single user	MS	$389
Source - **Multiuser**	MS	$799
dBx - translator	MS	$299
w/source to library	MS	$349

FEATURE

UI Programmer - Quickly generate dBASE User Interfaces, prototypes. Use supplied templates or create own. Pop-up help, bounce bar menus, screen forms. II, III, FoxBASE+, Quicksilver, Clipper PC $249

We support MSDOS (not just compatibles), PCDOS, Xenix-86, CPM-80, Macintosh, Atari ST, and Amiga.

CIRCLE 44 ON READER SERVICE CARD

THE PROGRAMMER'S SHOP

provides complete information, advice, guarantees and every product for Microcomputer Programming.

What Every Hard Disk User Needs . . .

Everyone with a fixed disk needs a set of reliable utilities that are more than just tools. The products at the right can increase your productivity by protecting your data, maintaining your hard disk, and making DOS directories easy to use. We use these, tools in-house, and they really work. GUARANTEED.

Order before July 31, 1987 and mention this ad for these SPECIAL PRICES:

	List	Normal	SPECIAL
Disk Technician - smart disk upkeep	$ 99	$ 89	**$ 79**
Xtree - classic graphic tree	$ 50	$ 45	**$ 35**
Back-It - flexible, fast	$100	$ 89	**$ 69**

C Support-Systems

Advantage C++	PC	$479
C Sharp - **realtime,** tasks.	PC	$495
C ToolSet - DIFF, xref, source	MS	$ 89
The HAMMER by OES Systems	PC	$129
Lattice Text Utilities	MS	$ 89
Multi-C - multitasking	PC	$135
PC LINT-Checker. Amiga $89	MS	$ 99
Quickshell - script compiler	PC	$349
Pfantasy Pac - by Phoenix	PC	$799
PforCe++	PC	$259
Pre-C - Lint-Like	MS	$155
Sapiens V8 - 8M workspace	PC	$269
SECURITY LIB - Source $229	PC	$115
Timeslicer	PC	$265

C-Screens, Windows, Graphics

C Power Windows by Entelekon	PC	$109
C-Scape - capture Dan Bricklin	PC	$179
Curses by Lattice	PC	$ 89
dBASE Graphics for C	PC	$ 69
ESSENTIAL GRAPHICS - fast	PC	$185
GraphiC - new color version	PC	$285
Greenleaf Data Windows	PC	$159
w/source	PC	$299
LightWINDOWS/C-for Datalight C	PC	$ 79
Multi-Windows - use w/ Multi-C	PC	$295
Screen Ace Form Master	PC	$195
Vitamin C - screen I/O	PC	$159
Windows for C - fast	PC	$189
Windows for Data - validation	PC	$319
View Manager - by Blaise	PC	$179
ZView - screen generator	MS	$169

Debuggers

386 Debug - by Phar Lap	PC	$129
Breakout - by Essential	PC	$ 89
CODESMITH - visual	PC	$ 99
C SPRITE - data structures	PC	$129
DSD87 - by Soft Advances	PC	$ 79
Periscope I	PC	$289
Periscope II	PC	$139
Periscope II-X	PC	$105
Pfix-86 Plus - by Phoenix	PC	$229
Turbo TDebug	PC	$ 55
Showcase - test software	PC	$125
SoftProbe II - embedded systems	PC	$695

FEATURE

I/O Pro - FORTRAN screen development system includes text/graphics editor, screen, string, graphics toolkit, communications, keyboard. Color support. IBM, MS, R-M, Supersoft FORTRAN. PC $129

Fortran & Supporting

50:More FORTRAN	PC	$ 99
ACS Time Series	MS	$399
Forlib+ by Alpha	MS	$ 59
MACFortran by Microsoft	MAC	$229
MS Fortran - 4.0, full 77'	MS	$279
No Limit - Fortran Scientific	PC	$115
PC-Fortran Tools - xref, pprint	PC	$165
RM/Fortran	MS	Call
Scientific Subroutines - Matrix	MS	$139

Multilanguage Support

BTRIEVE ISAM	MS	$185
BTRIEVE/N-multiuser	MS	$455
Flash-Up Windows	PC	$ 79
GSS Graphics Dev't Toolkit	PC	$375
HALO Graphics	PC	$205
Development Package	MS	$389
Informix 4GL-application builder	PC	$789
Informix SQL - ANSI standard	PC	$639
Opt Tech Sort - sort, merge	MS	$119
PANEL	MS	$215
Pfinish - by Phoenix	MS	$229
PolyLibrarian by Polytron	MS	$ 79
PolyBoost - speed I/O, keyboard	PC	$ 69
PVCS Corporate-source control	MS	$309
PVCS Personal	MS	$109
QMake by Quilt Co.	MS	$ 79
Report Option - for Xtrieve	MS	$109
Screen Machine	PC	$ 59
Screen Sculptor	PC	$ 95
SRMS - source control	MS	$109
Synergy - create user interfaces	MS	$375
VXM - multi-env. link	MS	$195
Xtrieve - organize database	MS	$199
ZAP Communications - VT 100	PC	$ 89

Pascal and Supporting

ALICE - learn Pascal	PC	$ **49**
Exec - Chain Programs	MS	$ 79
MetaWINDOWS-graphics toolkit		
bit-mapped, fast	PC	$115
MetaWINDOWS PLUS	PC	$185
Microsoft PASCAL - faster	MS	$189
Programmer's Extender	MAC	$ 79
Pascal Pac with Tidy	PC	$ 49
Pascal Tools PLUS	PC	$119
Pascal 2 - by Oregon Software	MS	$329
Turbo Extender by Turbo Power	PC	$ 75
TurboHALO - 150 routines	PC	$ 99
Turbo Optimizer	PC	$ 69

DataBase & File Management

DataFlex by Data Access	PC	$ 899
Dataflex multiuser	PC	$1149

RECENT DISCOVERY

SSP/PC - Fast math subroutine library in C and Assembler for C, Fortran, Pascal, and BASIC. 145+ routines include trig, elementary, hyperbolic, and gamma; chi square, polynomials, more. 8087 Support. PC $ 269

Other Languages

APL*PLUS/PC	PC	$ 429
CCS Mumps - Singleuser	PC	$ 50
CCS Mumps - Multiuser	PC	$ 359
MasterForth - Forth '83	MAC or PC	$ 109
Microsoft MASM	MS	$ 98
Modula-2 - by Pecan	MS	$ 79
Modula-2/86 by Logitech	PC	$ 62
Pasm - by Phoenix	MS	$ 115
PC Forth+ - by Lab Micro	PC	$ 199
Smalltalk/V	MS	$ 85
SNOBOL4+ - great for strings	MS	$ 80
UR/Forth	MS	$ 279

Xenix/Unix

Basic - by Microsoft	$ 239
C-Terp by Gimpel Software	$ 379
Cobol - by Microsoft	$ 639
Cobol Tools - by Microsoft	$ 319
Fortran or Pascal - by Microsoft	$ 439
MicroFocus Lev. II Compact COBOL	$ 795
Panel	$ 535
RM/Cobol	Call
RM/Fortran	Call
Xenix Complete System	$ 999
Xenix Development System	$ 499

Other Products

386 Assembler/Linker	PC	$ 389
Advantage Link	PC	$ 359
ASMLIB - 170+ routines	PC	$ 125
asmTREE - B+ tree file mgmt.	PC	$ 339
Compact Source Print	PC	$ 45
Dan Bricklin's Demo Program	PC	$ 59
Help/Control - on line help	PC	$ 99
Interactive Easyflow-HavenTree	PC	$ 125
Link & Locate - tools to work with		
Intel and Tektronix projects.	MS	$ 329
LMK - like UNIX make	MS	$ 139
Microsoft Windows	PC	$ 69
Software Development Kit	PC	$ 319
MKS Toolkit - Unix, vi, awk	PC	$ 99
Norton Commander	PC	$ 55
Numerical Analyst by Magus	PC	$ 269
PDisk - cache, tree	PC	$ 99
PLink - 86 PLUS - overlays	MS	$ 299
Polymake by Polytron	MS	$ 119
PolyShell by Polytron	MS	$ 119
PolyXREF by Polytron	PC	$ 99
Quelo 68000 X-ASM	PC	$ 509
Sapiens V8 - 8M virtual mgr.	PC	$ 269
Taskview - ten tasks	PC	$ 55
Tom Rettig's Library - dBASE	PC	$ 89
Tree Diagrammer	PC	$ 45
Visible Computer: 8088	PC	$ 65

Note: Mention this ad. Some prices are specials. Ask about COD and POs. Formats: 3" laptop now available, plus 200 others. UPS surface shipping add $3/item. All prices subject to change without notice.

Call for a catalog, literature, advice and service you can trust

VISA HOURS MasterCard

8:30 AM - 8:00 PM EST.

800-421-8006

THE PROGRAMMER'S SHOP™

Your complete source for software, services and answers

5-L Pond Park Road, Hingham, MA 02043

Mass: 800-442-8070 or 617-740-2510 5/87

"I like the way you do business. I ordered two programming packages from you and before I had time to wonder when I would receive them, they were on my desk. Good prices and fast delivery, - not a bad way to do business. - Thanks"

Jeff Schropfer
Bytrek

CIRCLE 45 ON READER SERVICE CARD

10 Important Reasons C Programmers Use Our File Manager

1. It's written in C.
Clearly the growing language of choice for applications that are fast, portable and efficient. All of db_VISTA's source code is written in C.

2. It's fast – almost 3 times faster than a leading competitor.
Fast access that comes from the unique combination of the B-tree indexing method and the "network" or direct "set" relationships between records. A winning combination for fast performance.

3. It's flexible.
Because of db_VISTA's combination of access methods, you can program to your application needs with ultimate design flexibility. Use db_VISTA as an ISAM file manager or to design database applications. You decide how to optimize run-time performance. No other tool gives you this flexibility without sacrificing performance.
db_VISTA is also well behaved to work with most any other C libraries!

4. It's portable.
db_VISTA operates on most popular computers and operating systems like UNIX, MS-DOS and VMS. You can write applications for micros, minis, or even mainframes.

5. Complete Source Code available.
We make our entire C Source Code available so you can optimize performance or port to new environments yourself.

6. It uses space efficiently.
db_VISTA lets you precisely define relationships to minimize redundant data. It is non-RAM resident; only those functions necessary for operation become part of the run-time program.

7. Royalty free run-time.
Whether you're developing applications for yourself or for thousands, you pay for db_VISTA or db_QUERY only once. If you currently pay royalties to someone else for your hard work, isn't it time you switched to royalty-free db_VISTA?

db_VISTA™

Features
- **Multi-user** support allows flexibility to run on local area networks
- **File structure** is based on the B-tree indexing method
- **Transaction processing** assures multi-user consistency
- **File locking** support provides read and write locks
- **SQL-based db_QUERY** is linkable
- **File transfer** utilities included for ASCII, dBASE optional
- **Royalty-free** run-time distribution
- **Source Code** available
- **Data Definition Language** for specifying the content and organization of your files
- **Interactive database access** utility
- **Database consistency check** utility

File Management Record and File Sizes
- Maximum record length limited only by accessible RAM
- Maximum records per file is 16,777,215
- Maximum file size limited only by available disk storage
- Maximum of 256 index and data files
- Key length maximum 246 bytes
- No limit on number of key fields per record
- No limit on maximum number of fields per record

Operating System & Compiler Support
- **Operating systems:** MS-DOS, PC-DOS, UNIX, XENIX, UNOS, ULTRIX, Microport, VMS
- **C compilers:** Lattice, Microsoft, IBM, DeSmet, Aztec, Computer Innovations, Turbo C, XENIX and UNIX

8. SQL-based db_QUERY™
Add our new C-linkable, SQL-based, ad hoc query and report-writing companion product to provide a simple relational view of your db_VISTA applications. Without compromising speed.

9. Free tech support.
60 days of free technical and application development support for every Raima product. Of course, extended support and training classes are also available at your place or ours.

10. Upward database compatibility
Start out with file management in a single-user PC environment—then move up to a multi-user LAN or a VAX database application with millions of records. You'll still be using db_VISTA. That's why so many C programmers are choosing db_VISTA.

But don't just take our word for it.
"Raima's customer support and documentation are excellent. Source code availability and royalty-free run-time is a big plus."
Dave Schmitt, President
Lattice, Inc.

"db_VISTA has proved to be an all-round high performer in terms of fast execution, flexibility and portability, and has undoubtedly saved us much time and development effort."
John Adelus, Hewlett-Packard
Office Productivity Division

30-day Money Back Guarantee!
Try db_VISTA in your environment for 30 days and prove it to yourself. If not completely satisfied, return it for a full refund.

Price Schedule

	db_VISTA	db_QUERY
☐ Single user	$ 195	$ 195
☐ Single user w/Source	$ 495	$ 495
☐ Multi-user	$ 495	$ 495
☐ Multi-user w/Source	$ 990	$ 990
NEW:		
☐ VAX Multi-user	$ 990	$ 990
☐ VAX Multi-user w/Source	$1980	$1980

Order Now.
Put db_VISTA to work in your application program. Ordering is easy—simply call toll-free. We'll answer your technical questions and get you started. Call today.

Call Toll-Free Today!
1 (800) db-RAIMA
(800/327-2462) or
206/828-4636

MasterCard VISA

RAIMA™ CORPORATION

3055 - 112th NE, Bellevue, WA 98004 USA
(206) 828-4636 Telex: 6503018237 MCI UW

CL 7/87

DESIGNING WITH DATA BASES

Fidelity and the full relational model

By Matthew Rapaport

Back in March, I began by listing 14 qualities, cited by James Martin, that make a language fourth generation. Since that time, this column has focused on the third element of Martin's list: possession of an inherent data management system.

In April we discussed some general data models and the ANSI Standard Planning and Requirements Committee (SPARC) three-schema architecture ("DBMSs in the real world," *COMPUTER LANGUAGE*, pp. 95-100). E.F. Codd's relational model[1] was the source for the examples and illustrations in our discussion of directories and data dictionaries ("Directory and dictionary components," *COMPUTER LANGUAGE*, May 1987, pp. 79-84), and integrity issues ("Data base integrity," *COMPUTER LANGUAGE*, June 1987, pp. 87-94).

Some of the integrity issues we've discussed (like the entity and referential integrity rules) apply only to relational systems, while others (for example the concept of *DOMAINS*) can be usefully applied to hierarchical, network, and other data models.

I did, however, address these subjects from a relational viewpoint because most DBMS products in either small (micro) or medium-size (multiuser micro and mini) computer systems claim to be relational. The only exception I know of (and I am including here only those products that manage multiple files/tables at the same time) is Micro Data Base Systems' MDBS III, a powerful, package based on the CODASYL network model.

DBMSs have been around much longer than fourth generation languages. I have addressed only a few DBMS design and usage aspects, and I will explore these and other issues again as we discuss using DBMS and fourth generation products in application development in a more formal way. I will also briefly discuss other aspects of the fourth generation language environment: procedural and nonprocedural languages, screen and report generators, and styles of integration between various components. First, however, let's evaluate some of the popular products on the market using Codd's criterion for a relational DBMS.

In October of 1985, articles written by Codd appeared in two consecutive issues of *Computerworld*.[2,3] In these articles he delineated 12 rules, 18 manipulation features, nine structural features, and three integrity features that make a DBMS fully relational.

In these now famous articles, Codd compared three well-known mainframe-based DBMSs for compliance with these rules. Few of us have to decide whether to buy IBM's Information Management System or Cullinet, products which cost around a quarter of a million dollars. More likely, we may choose between Ashton-Tate's dBASE, Microrim's R:base, Unify Corp.'s UNIFY, Informix Software's Informix, and other micro or super-micro based products. I will therefore review Codd's rules and attempt to apply a variation of his qualification scheme to the products I have been using as examples.

We find that compliance with the rules as a whole does not necessarily mean a product is fit or unfit for any given DBMS application. Different applications are best met by tools with strengths in areas that are important to a particular use, even if they are in other areas.

In the last issue, I alluded to a general difference between management information systems (MIS) and Data Processing (DP) applications. This distinction, along with others such as the scope of a given application, strongly influence product choice. I will look at these differences in some detail, bearing in mind that the degree of compliance with Codd's rules is only one factor in the decision to commit development to a particular DBMS.

To be minimally relational, a DBMS must explicitly represent data in tables and not require that any data be embodied in pointers, links, or ordering. A second requirement is for the data manipulation language (DML) to support the basic relational operations of projection, selection, and one of several kinds of join operations.

Products that do not support both of these requirements may be relational like, but not true relational data bases. As we shall see, many of the most popular DBMSs do not meet both of these criterion. Often their failure is due to an implicit consequence of the minimalist rules we shall explore later. But first, the rules.

1. Information rule. All data in the data base is represented explicitly as values in tables (not in links, pointers, or ordering). This rule is the basic foundation of the relational model, and the first part of the minimal rule.

2. Guaranteed access rule. Every datum in the data base is guaranteed to be accessible through its own primary key. This rule is also part of the relational model's foundation. Supporting techniques vary widely among products. Ideally, key uniqueness is supported at the data dictionary or directory level of the DBMS and not through indexes, though the latter will satisfy the rule if they cannot be subverted—access must be forced through the index for all kinds of applications and languages that can utilize the data.

3. Systematic treatment of *NULL* values. This seemingly innocuous rule has spawned a body of DBMS theory.[4-6] The underlying problem is that the nonexistence of something is impossible to represent by the presence of some marker that does exist without potential contradictions.

For example, consider an employee whose skills are unknown, represented by the presence of N/A (Not Applicable) or a question mark (?) in the skill attribute position. It is difficult to find a single way to handle all possible variations of a query like "Show me everyone who has skill *X*" and "Show me everyone who does not have skill *X*" and always produce meaningful answers. Sometimes we want nulls to be counted as belonging to a select set, and other times we do not.

Compliance with this rule is often found in extensions to the DBMS query language such that the user can switch the different senses of *NULL* on or off.

4. Dynamic on-line catalog rule. The

data base dictionary that describes the data must also represent its metadata in tabular format with all of the integrity and manipulative characteristics of the data base as a whole. This means that the dictionary can be created, queried, and updated with the same DML as the data itself.

5. Comprehensive data sublanguage. Whatever languages the DBMS may support, there must be at least one language whose statements are expressible as formally specified strings and that supports all of the following:

- Data definition (creating base relations)
- View definition (external views)
- Data manipulation (select, project, and join)
- Establishing integrity constraints
- Establishing security barriers
- Delineating transaction boundaries (begin, commit, and rollback).

Part of this rule comprises the second half of the minimalist rule. Products that seemingly fail the minimalist rule often do so because they do not support true joins at the set level. Otherwise powerful products like Cosmos' Revelation and DataAccess Corp.'s DataFlex do not support an operation that logically joins two tables to create a third table. While many of the products mentioned in this column do support this operation, most of these products would fail the other requirements of rule 5.

6. View-updating rule. This rule states that the system can update any view that theoretically can be updated. A view can be updated if a time-independent algorithm changes the view in precisely the desired way when applied to the base relations that compose that view.

Here, update also refers to inserts and deletes. This is not a trivial feat, as you can see if you look back at Figure 2 in my April column. If, for example, the address of Customer 1 in the left-most base table were to change, four tuples in the external view would instantly have to reflect this change to whatever application software happened to be using it at the moment.

7. High-level language rule. The comprehensive data sublanguage specified in rule 5 must also be able to handle both base and derived relations as single operands, that is, sets of rows in a single operation involving either retrieval, insertion, or updating of data. The dBASE statement: *REPLACE ALL AMOUNT.DUE WITH AMOUNT .DUE*1.05 WHERE DAYS.DUE > 90* is a typical example of the use of this kind of high-level language statement. This statement produces a new relation: *(ALL TUPLES WITH DAYS.DUE > 90)* and, from the user's perspective, performs a single operation at the same time.

8. Physical data independence. This rule demands that applications code using a DBMS remain logically unimpaired if physical changes are made to the files that constitute the data base. Such changes include moving a file from one disk volume to another, splitting a large file into two or more files, physically joining two files into one, changing the data structure of a physical file, or any performance alterations (sorting, setting up new indexes, recreating hash tables, and other operations not typically available to micro-based developers).

This rule does not insist that such applications be immediately usable after physical file changes (recompilation may be required, for example), only that their logic not be compromised by physical alterations to accessed files.

One of the primary reasons for the development of DBMS systems was to satisfy this rule. The rule is usually waived during an attribute deletion, but it is possible to envision ways to automatically shield applications logic from deletions. To date, no product supports automatic delete protection.

9. Logical data independence. Like rule 8, this rule applies to the logic of applications software, which should remain unimpaired by changes to the conceptual level of the data base (see my April column for a discussion of architectures). Such changes include splitting a relation in two, joining two or more relations, adding attributes, and any other changes that theoretically permit unimpairment.

Again, deleting an attribute is outside this requirement, as are operations like changing attribute lengths or data types. These changes may or may not accompany physical changes to the data files. Note also that this rule is more restricted

Codd's relational criteria

Structural features

S1. Relations of assorted degrees. Tables with various numbers of attributes, unnumbered rows (no meaning in order), no positional concepts, named columns, and no repeating groups.

S2. Data wholly represented in tables.

S3. The result of a query is another table, which may be treated like any other table—a query table.

S4. View, or virtual, tables. External views that have internal representation in the commands and definitions that create them but do not store any data.

S5. Snapshot tables, evaluated (created) and stored along with some directory entry indicating the date and time of creation.

S6. Every column of every table is an attribute—no pointers per se.

S7. Each attribute obtains its values from a real domain.

S8. In each row, there is at least one combination of attributes whose values identify that row unambiguously—a primary key.

S9. Attributes can reference each other by meaning. This occurs when an attribute in one table is on the same domain as the primary key of another table. A foreign key can be thought of as an ambassador from the head-of-state (primary key) in one relation to some other relation.

Integrity features

I1. Entity integrity (covered at some length in last month's column).

I2. Referential integrity (also discussed last month).

I3. User-defined integrity. Essentially, the other mechanisms involved in what I termed class 1 integrity issues in "DBMSs in the real world," *COMPUTER LANGUAGE*, April 1987, pp. 95-100. These include masks, range tests, triggered procedures, and enumerated types.

Manipulative features

M1. Theta select
M2. Project
M3. Theta join
M4. Outer theta join
M5. Divide
M6. Union
M7. Intersection
M8. Set difference
M9. Outer union
M10. Relational assignment
M11. Theta select maybe
M12. Theta join maybe
M13. Outer theta join maybe
M14. Divide maybe
M15. Theta select semantic override (s/o)
M16. Theta join s/o
M17. Outer theta join s/o
M18. Divide s/o

Theta refers to the algebraic operations >, <, =, <>, etc. The maybes and the semantic override are extensions made necessary by the problems arising from the need to treat *NULLS*.

Software designed for IBM PC and Compatibles.

Lattice® C Compiler — *$500.00*

With more than 30,000 users worldwide, Lattice C Compilers set the industry standard for MS-DOS and PC-DOS computers. Lattice C gives you all you need for development of programs on the Intel iAPX86 processor family. Lattice C is a full implementation of Kernighan and Ritchie with the ANSI C extensions and many additional features for MS-DOS and PC-DOS.

C-SPRITE™ — *$175.00*

Simplify debugging of programs written in Lattice C and most other languages. C-SPRITE is the first software tool for *both* source level and machine level debugging. It gives you a high-level view of C programs with function names, source lines, variable names and C data types. In addition, C-SPRITE provides a low-level view of machine addresses and instructions. Debugging through a communication port, help screens, macros, command files, conditional commands and support for PLINK86 overlays are also provided.

Lattice Screen Editor (LSE™) — *$125.00*

Designed as a programmer's editor, LSE is fast, flexible and easy to learn. LSE's multi-window environment provides all the editor functions you need including block moves, pattern searches and "cut and paste." In addition, LSE offers special features for programmers such as an error tracking mode and three Assembly Language input modes. You can also create macros or customize keystrokes, menus, and prompts to your style and preferences.

dBC III™ — *$250.00*

The dBC III library lets you create, access and update files that are compatible with Ashton-Tate's dBASE system. dBC III's C functions let you extend existing dBASE applications or allow your users to process their data using dBC III or dBASE III.

LMK™ — *$195.00*

An automated product generation utility compatible with UNIX Make, LMK lets you rebuild complex programs with a single command. Once you specify the relationships of the various pieces of your system in a dependency file, LMK automatically rebuilds your system the same way every time, and only compiles program files that have changed. But LMK is not limited to updating programs. You can use LMK to update documentation or perform **any** executable command!

Curses™ — *$125.00*

The Lattice Curses screen interface is a library of 84 C functions used for updating the screen of an IBM PC. It is compatible with UNIX curses packages, and can be used to transfer UNIX curses programs to the PC. Lattice Curses can keep and update any number of full or partial virtual screen images in memory and paint the actual screen as needed with one or more of the images. Functions are provided to write text, move cursors, scroll, overlay, outline, insert, delete, clear and highlight the virtual screens.

SideTalk™ — *$119.95*

No more interruptions! Because now SideTalk can transfer your files or receive your mail while you're using your computer for other important matters. And SideTalk is programmable with its own BASIC-like language. What's more, you're never more than a keystroke away from all of the power of SideTalk.

Text Management Utilities (TMU™) — *$120.00*

TMU consists of eight software tools to help you manage your text files. **GREP** searches files for the specified pattern. **DIFF** compares two files and lists their differences. **EXTRACT** creates a list of file names to be extracted from the current directory. **BUILD** creates batch files from a previously generated file name list. **WC** displays the number of characters and optionally the checksum of a specified file. **ED** is a line editor which can utilize output from other TMU software in an automated batch mode. **SPLAT** searches files for a specified character string and replaces every occurrence with a specified string. And **FILES** lists, copies, erases or removes files or entire directory structures which meet the specified conditions.

TMU is not system or language oriented. You can use it on files created with a text editor or produced by a text processor. Files written in C, Assembly, COBOL, Fortran, Pascal or English can all be handled by TMU.

Panel™ — *$295.00*

Panel will help you write your screen programs and layer your screen designs with up to ten overlapping images. Panel's screen layouts can be assigned to individual windows and may be dynamically loaded from files or compiled into a program. Panel will output C source for including in your applications. A monitor and keyboard utility is also included to allow you to customize your applications for other systems.

SecretDisk® — *$119.95*

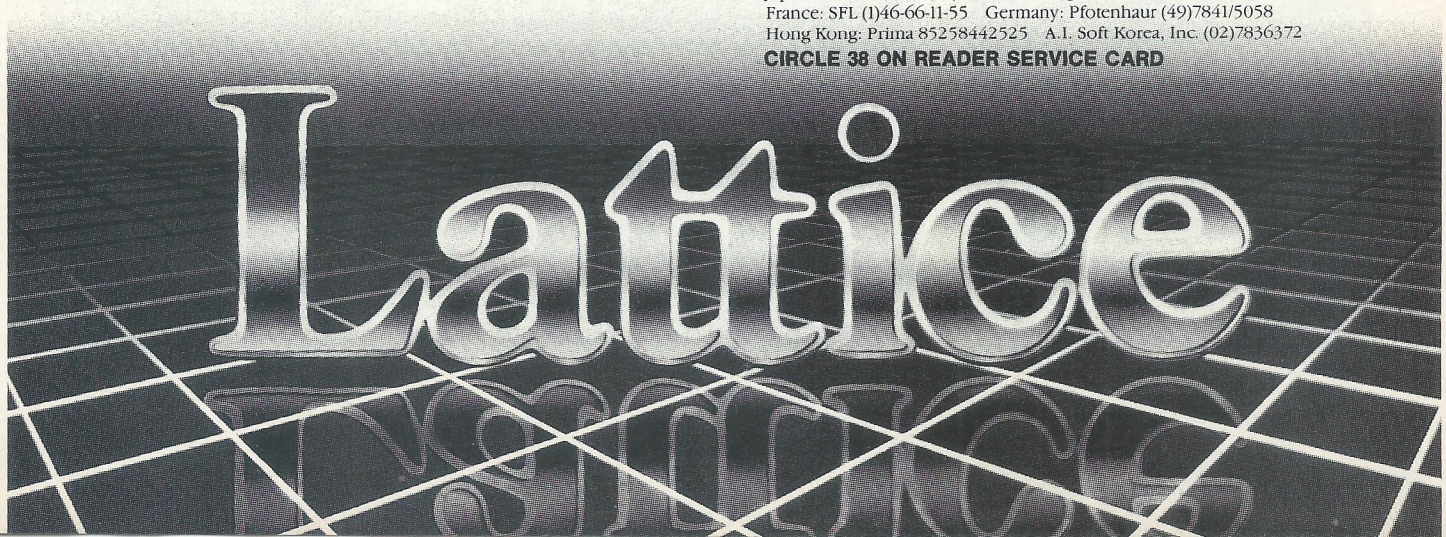
SecretDisk gives you automatic data encryption for your sensitive files. SecretDisk is loaded as a disk driver by MS-DOS. It creates new "SecretDisk drives" on your floppy or hard disk where *all* data and programs are *always* fully encrypted. SecretDisk is easy to use. You get the convenience of a password system with the full protection of the NBS Data Encryption Standard. You enter your password when the system is booted, and switch protection on and off with a single password-controlled command line. Not for export.

Lattice

Lattice, Incorporated
Post Office Box 3072
Glen Ellyn, Illinois 60138
(312) 858-7950 TWX 910-291-2190

INTERNATIONAL SALES OFFICES: Benelux: Ines Datacom (32) 2-720-51-61
Japan: Lifeboat, Inc. (03)293-4711 England: Roundhill (0672)54675
France: SFL (1)46-66-11-55 Germany: Pfotenhaur (49)7841/5058
Hong Kong: Prima 85258442525 A.I. Soft Korea, Inc. (02)7836372

CIRCLE 38 ON READER SERVICE CARD



than rule 8, which does not qualify a class of changes to which the rule applies.

Where physical files are concerned, any change (except deletion) should remain invisible to the application program (and, by implication, to the conceptual schema). However, the conceptual level is one step nearer the application, and applications logic becomes sensitive to a broader class of alterations.

The inability of early data models to support this feature was one of the reasons for the development of the relational model. But even a DBMS that is relational in the sense that it supports tables and references by meaning will fail this requirement unless it also supports external views that can be updated.

External views are the third schema of the ANSI SPARC architecture. Codd appears to have anticipated the findings of ANSI SPARC in demanding that view support (rule 6) be included in the full concept of the relational model.

10. Integrity independence. A substantial portion of my last column was spent on part of this rule, which includes both entity and referential integrity rules. The entity integrity rule imposes strict format standards on primary keys of relations.

Specifically, each key must be unique, indentifying each tuple in a relation purely by reference, without regard to pointers, offsets, or information embodied in any formal link.

Second, every component of the key must be necessary to establish this uniqueness when the relation is seen as a whole.

The referential rule extends the power of the entity rule to all the relations of a data base. Each time the primary key domain of one relation is named in an attribute of another relation, or foreign key, (as when the *employee* relation names the *job* relation with the attribute *job.number*) that particular key must already exist in the set of primary keys actually in the named relation. This ensures that only one tuple will be returned when one relation points (by reference) at another. It also indirectly guarantees that every tuple in the data base is not only accessible without use of traditional pointers, but that no foreign key misleads by referencing a primary key (and tuple) that doesn't exist.

More than this though, Codd specifies that whatever kinds of integrity constraints are applied to data, they must be specifiable in the data sublanguage of rule 5 and be storable in the system catalog (the directory/dictionary), not in applications software. Since there are many integrity constraints, no product can satisfy this rule (even if it supports entity and referential integrity at the directory level) unless it has a dictionary of uncommon power.

11. Distribution independence. Like the physical independence rule, distribution independence requires that applications programs remain logically unimpaired if the physical files of the DBMS become distributed across multiple CPUs. It is not necessary that a particular product actually support distributed data bases to qualify under this rule, only that their associated data manipulation language theoretically be able to function unimpaired if the data could be distributed.

12. Nonsubversion. This last rule deals with various language systems that may be able to access and update data. It states that the controlling DBMS must not permit any interface that can subvert the integrity protection provided by the DBMS to the attributes of any relation.

This protection extends to all attributes in their respective domains and is especially important to primary and foreign keys. A COBOL program cannot be allowed to place a value in a foreign key if that value does not already exist in the relation to which the key refers. A BASIC program cannot put a string like *4153971881* in an attribute if the format of that attribute is supposed to be *(999) 999-9999*.

Obviously, this rule cannot be supported if the data dictionary does not directly control most of the integrity checking.

* Besides the 12 main rules, Codd delineated nine structural, 18 manipulative, and three integrity features that must all be supported fully for a DBMS to claim full fidelity to the relational model. He combined these features and rules, 42 in all, and gave each product one point for each rule or feature it fully supported. Doubling the resultant score yields an approximate fidelity percentage, and Codd recommended that this figure be rounded to the nearest significant 10%. The highest score was 46 (roughly 50%) for

Rule features

Features	Related rules
S1	
S2	1
S6	
S3	
S4	6
S5	
S7	See manipulation group
S8	
S9	2, 10
M1 ⋮ M14 ⋮ M18	3, S7 → 7
I1	
I2	10
I3	

Codd's 30 features in association with their corresponding rules.

Table 1.

Rating products by Codd's rules

Products	Rules 1 Information	2 Access	3 Nulls	4 Dictionary	5 Data sublanguage	6 Views	7 HLL	8 Physical indep.	9 Logical indep.	10 Integrity	11 Distributive indep.	12 Non-subversion	T	%
dBase III+	2	2	0	0	1	2	2	0	0	0	0	0	9	38
R:base Sys. V	2	2	0	0	0	2	2	0	1	1	0	2	12	50
Revelation	2	2	0	2	0	0	0	0	0	2	0	0	8	33
Supra	2	2	0	1	0	2	1	2	2	1	2	2	17	71
Unify	2	2	0	0	0	0	2	1	0	2	2	1	11	46
Informix	2	2	2	0	2	2	2	1	1	0	2	0	16	67
Dataflex	2	2	2	0	0	0	0	1	0	0	2	0	9	38
Paradox	2	2	0	1	0	1	1	0	0	1	0	2	10	42
Nomad 2	2	2	0	0	0	0	2	1	1	2	0	0	10	42

A "0" indicates no support for the corresponding rule. A "1" indicates partial support in one form or another, while a "2" signifies relatively full support for the rule.
Factor = 4.17.

Table 2.

IBM's DB2.

Codd's approach to the ratings is not as informative as it could be. The 30 extra features have varying degrees of relation to the 12 main rules. Sometimes two or more features comprise the essence of a main rule, other times they extend but do not replace the main rules. The rules are unevenly represented in the features; some associate with many features, others with few or none, and one of the features is not clearly associated with any rule.

The accompanying sidebar, "Codd's relational criteria," sketches the 30 features. I will look at these in greater detail next month. Table 1 groups the features by the rules they seem to fit best.

Because some products seem to partially support some of the rules, I thought it appropriate to provide a category of partial qualification. Support for each of the rules is scored with a 0 for no support, 1 for partial support and 2 for full support. The results of some of the well-known products are shown in Table 2. The best possible score is be 120, so I scaled each of the scores by 4.17 to derive the percentage figures shown in the right-most column.

The evaluation presented in Table 2 is arguable because partial qualification (a score of 1) depends upon my observation that a product satisfies some but not all parts of a given rule. Partial support does not imply that a product supports half of a rule's implications, only that it does not satisfy the rule fully but obviously contains some support.

In the next issue we will analyze these rules in association with the features and discuss the value of this analysis in the selection of a DBMS product for real applications.

References

1. Codd, E.F. "A Relational Model of Data for Large Shared Data Banks." *Communications of the ACM*, vol. 13, no. 6 (June 1970):377-387.
2. Codd, E.F. "Does your DBMS run by the rules." *Computerworld* (Oct. 14, 1985):ID1-ID9.
3. Codd, E.F. "Does your DBMS run by the rules." *Computerworld* (Oct. 21, 1985):49-60.
4. Date, C.J. *An Introduction to Database Systems*, vol. 2, Addison-Wesley (1985).
5. Lipski, W. Jr. "On Semantic Issues Connected with Incomplete Information Databases," *ACM Transactions on Database Systems* vol. 4, no. 3 (Sept. 1979).
6. Zaniolo, C. "Database Relations with Null Values," *Journal of Computer and System Sciences*, vol. 28, no. 1 (Feb. 1984):142-166.

Matthew Rapaport is currently employed by AMEX Life Assurance in San Rafael, Calif., and has worked in fourth generation language application environment development for six years.

NO ROYALTIES

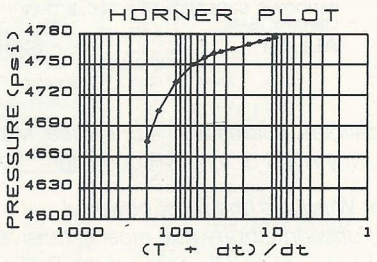
GRAPHICS TOOLS for C and FORTRAN-77 PROGRAMMERS

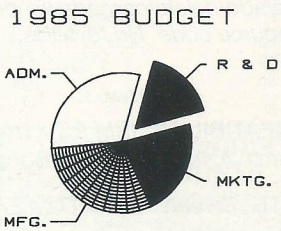
GRAFLIB NEW VERSION 2.00 $175.00

Screen graphics library for IBM-PC and compatibles.
- Supports CGA, EGA and HERCULES Graphics cards.
- Routines to draw lines, arcs and circles.
- Smooth curve fitting routines (cubic splines).
- Text labeling in 9 different orientations.
- Feature to produce hardcopy of graphics screen on Epson compatible printers.
- Routines to SAVE and RECALL graphics screen.

PLOTHP NEW VERSION 2.00 $175.00

Plotting routines for Hewlett-Packard and HPGL compatible plotters. Some of the features are:
- Routines to plot lines, arcs, circles and ellipses.
- Routines to draw smooth curves through discrete data points (using cubic splines).
- Routines to scale and plot Linear, Semi-log and Log/Log plots.
- Text labeling in various directions and 9 different orientations.





PLOTHI NEW VERSION 2.00 $175.00

Plotting routines for Houston instrument and DM/PL compatible plotters. Some of the features are:
- Routines to plot lines, arcs, circles and ellipses.
- Routines to draw smooth curves through discrete data points (using cubic splines).
- Routines to scale and plot Linear, Semi-log and Log/Log plots.
- Text labeling in various directions and 9 different orientations.

FORTLIB NEW VERSION 2.00 $125.00

Enhancement routines for FORTRAN-77 (PC-DOS/MS-DOS).
- RS-232 communication routines.
- Key board, video and disk handling routines.
- Sound and primitive graphics generation routines.
- And many more miscellaneous feature packed routines.

Some of the compilers supported are:

C	FORTRAN-77
CI-C86	DR FORTRAN-77
Lattice C	IBM Professional Fortran
Microsoft C	Lahey F77L
MARK WILLIAMS C	Microsoft Fortran
	RM Fortran

For additional information please call or write.

Sutrasoft (formerly The Librarian, Inc.)
P.O. Box 1733
Sugar Land, TX 77487-1733
(713) 491-2088

Source Code Included

CIRCLE 47 ON READER SERVICE CARD

C POWER

FREE! Turbo C® (Borland)

(with C Business Library or C Starter Package until 7/31/87.)

POWER WINDOWS
FUNCTION LIBRARIES
SUPERFONTS FOR C
B-TREE & ISAM
C-TERP
SCREEN PAINTER
& FORMS CREATOR

POWER WINDOWS

MOST POWERFUL YET
POP-UP WINDOWS FOR
Menus/Overlays
Help Screens
Messages/Alarms
ZAP ON/OFF SCREEN
FILE-WINDOW MANAGEMENT
AUTOMATIC
Horizontal & Vertical Scrolling
Word Wrap & Line Insertion
Cursor/Attributes/Borders

The most powerful, flexible and easy to use windowing package available! Many types of menus. Highlighting. Move data between files, keyboard, program and windows. Status lines. Change size/location/overlapping. Move/add/delete/cascade windows. Full source code. No royalties.
3 disks $129.95

C FUNCTION LIBRARY

BEST YOU CAN GET
400 FUNCTIONS
FULLY TESTED
SUPERB DOCUMENTATION
BETTER FUNCTIONS

Most complete screen handling plus graphics; cursor/keyboard/data entry, 72 string functions with word wrap; status and control; utility/DOS BIOS/time/date functions; printer control & more. Special functions. Functions you NEED but don't have! Full source code. No royalties.
4 disks $129.95

C BUSINESS LIBRARY

INCLUDES C FUNCTION LIBRARY, POWER WINDOWS, SUPERFONTS FOR C, B-TREE LIBRARY, ISAM
ALL for $299.95
(A $440 VALUE)

SUPERFONTS FOR C

SUPER SIZE CHARACTERS
Monochrome adapter
Color/graphics adapter
8 FONT LIBRARIES

Dramatic high impact screens/titles/messages. Use with or without windows. Great for projection TV. Create own font & image libraries or use ours. Use with windows for special effects or animation. Full source code. No royalties.
Font and Function Library . . .$49.95

B-TREE LIBRARY & ISAM DRIVER

POWERFUL DATA MANAGER
Fixed/Variable length records
FAST! EASY TO USE!
16.7 MILLION RECORDS/FILE
16.7 MILLION KEYS/FILE

Fast B-tree indices. Add/remove keys Find first/last/next/any key. Find keys by Boolean selection. Read/write/delete or add records to file. Full source. No royalties $129.95

C STARTER PACKAGE

INCLUDES C FUNCTION LIBRARY, POWER WINDOWS, SUPERFONTS FOR C
ALL for $199.95
(A $310 VALUE)

COMPATIBILITY: *IBM PC/XT/AT & compatibles. Microsoft C 4.0, Lattice, Mark Williams, Aztec, CI 86, Datalight, Ecosoft, any ANSI or full K&R compiler.*

DEPTH: *Screen Painter/Data Entry/Power Windows (the most powerful professional windowing package) and C Function Library (the most extensive and useful library available) plus the speed, power and flexibility of the B-TREE and ISAM library. All work together to make your project go FAST!*

NO MATTER WHAT ELSE YOU HAVE, GET THESE...
DON'T PAY MORE AND GET LESS!

C DYNAMO SCREEN PAINTER AND FORMS CREATOR

DOES IT ALL
RIGHT FROM YOUR KEYBOARD
AUTOMATIC CODE GENERATION
Data or Help Screens & Windows
Data Entry Screens & Windows
Menus
FAST, FLEXIBLE, EASY
Save Man-Months of Programming
Full Control of Screen Attributes
Monochrome or Color
by Form, Screen, Window or Item
FAST EASY MENU GENERATION
1-2-3 Like, Many Others

Requires Dynamo Data Entry
Screen painter & manual $129.95

DYNAMO DATA ENTRY

UNIQUE, POWERFUL, NECESSARY
Full Validation of Each Keystroke
Range Checking
Data Security to Item Level
Scrollable Data Entry Forms with Fixed & Scrollable Parts
Allows Forms Larger Than Screen
Over 40 Item Types
Powerful "Picture" Capability
Unique: Mix Text, Data Entry Fields With Menu Items

Full source code. No Royalties
Code plus manual $129.95

C-TERP

INTERPRETER FOR C
NO COMPROMISE, FULL K&R
BUILT IN SCREEN EDITOR
FAST, FAST COMPILE/LINK
USE EXTERNAL LIBRARIES!
SYMBOLIC DEBUGGING
SINGLE STEPPING
RAVE REVIEWS!

2 disks and manual. $299.95
With and library $199.95

C TOTAL LIBRARY

INCLUDES C FUNCTION LIBRARY, POWER WINDOWS, SUPERFONTS FOR C, B-TREE LIBRARY, ISAM AND C-TERP C interpreter
ALL for $499.95
(A $740 VALUE)

	LIST
PCNX™ UNIX WORKALIKE	$99
PCVMS™	99
O/S TOOL BOX™ (Build your own operating system)	99

™WENDIN SOFTWARE

THE C POWER FAMILY ... YOUR COMPLETE SOURCE FOR C

Entelekon™

SINCE 1982

12118 Kimberley, Houston, TX 77024 713-468-4412 VISA-MASTERCARD-CHECK-COD

CIRCLE 48 ON READER SERVICE CARD

EXOTIC LANGUAGE OF THE MONTH CLUB

Knowledge representation with CRL

By Doug Schuler

The description of reality has long challenged mankind. The conceptual metamorphosis that occurs from perception of the real world to a description, knowledge, or an efficient yet natural representation of knowledge in software is essentially a philosophical pursuit. Nevertheless, demands from the real world create an acute need for an approximate solution. In this modern age, knowledge representation is a central concern of artificial intelligence research. AI research is a quest for appropriate representations of complex objects and appropriate means of utilizing those representations.

Computer scientists and programmers must serve two masters. They must represent knowledge in a way that is sensible to users and to computers. Knowledge representation is an important concept for programmers of conventional software as well as AI programmers. Knowledge representation is an important component of present AI tools and will likely be introduced into conventional programming. Moreover, understanding the concepts of knowledge representation would be helpful to understanding the semantics of complicated knowledge in conventional programming.

Many researchers believe that the key to knowledge representation in computer software lies in the ability to gather attributes into a small set of similar objects that are manipulated with a small set of operations. One of the earliest developments in representing world knowledge in a flexible data structure for a computer was Marvin Minsky's concept of frames in 1975.[1] A frame is a data structure having some similarities to a Pascal record that possesses slots and values. Frames are differentiated from records by the variety of values, including functions, that slots can hold. Frames can also be connected into frame systems when other frames are referenced in slot values.

For example, a frame called *dog*, in which the pertinent attributes of a dog are stored, can have a slot called *ISA* and a value of *MAMMAL*. Values can be propagated or inherited throughout the frame system. In the dog example, a dog may inherit *WARMBLOODEDNESS* from the mammal frame. Frames are also used in a generic sense to describe similar data structures in various systems, such as schemas in the Carnegie Representation Language (CRL) or units in Intellicorp's Knowledge Engineering Environment (KEE).

Frames and framelike constructs are currently providing the foundation for much AI work. This research is often manifested in the development of representation languages. Earlier work in this field includes that of T. Winograd,[2] B. Kuipers,[3] and the Frame Representation Language group at the Massachusetts Institute of Technology, Cambridge, Mass.[4] Douglas Lenat and other researchers at Microelectronics and Computer Technology Institute are using frames in the CYC project,[5] an ambitious effort to represent all the knowledge found in a standard desk encyclopedia.

CRL is a frame language derived from the Schema Representation Language. SRL was developed at Carnegie-Mellon University, Pittsburgh, Pa.[6] for use in AI applications such as process planning and engineering project management.

CRL is the underlying glue of Knowledge Craft, a commercial product available from Carnegie Group Inc. for development of knowledge-based systems. Knowledge Craft is a sophisticated AI program development environment for industrial, financial, and other applications. Knowledge Craft provides a user-friendly interface to the system's schemas and capabilities, including editors and graphic representations of the system. Forward-chaining and backchaining languages (OPS-5 and PROLOG), object-oriented programming, demons, and event-driven programming are all supported. Knowledge Craft is an integrated collection of tools for knowledge-based programming in the same family as KEE and Inference Corp.'s ART. Although CRL is copyrighted by Carnegie Group Inc., it is basically the same as SRL, which is in the public domain. Therefore, many of my comments apply to both languages.

CRL should be used as a tool for solving problems that are not easily addressed through conventional programming. These problems include problem domains that are interrelated in complex ways and whose relations may change. CRL should be used if the solution to the problem seems just out of reach of conventional programming.

A frame in CRL is called a schema. CRL is essentially a collection of functions, written in Common LISP, that are used to create, modify, relate, test, and delete schemas, slots, and values. These schemas are related via relations and together form networks of knowledge. Let's look at a specific example to demonstrate these features.

For several years I have wanted a program that would help me decide what to do based on my assigned tasks and considering my current resources. I didn't want a machine telling me what to do—just one that would remember pertinent information and make intelligent suggestions. The program would do some scheduling, provide meaningful defaults, and have an acceptable user interface for entering, viewing, recording, and purging things to do. Of course, I would reserve the right to ignore its advice and do something else if I felt like it.

It seems clear that at the core of a Things-to-Do Advisor there must be a representation of a thing to do. We will start our system by defining one of these. The CRL function *CREATE-SCHEMA*, written in LISP, is used to create a schema called *THING-TO-DO*. When the schema is created, the name is returned:

```
> (CREATE-SCHEMA 'THING-TO-DO)
THING-TO-DO
```

Next, let's create some slots in the THING-TO-DO schema (Listing 1).

Presumably, any *THING-TO-DO* would have a priority, location where the task could be accomplished, due date, and resource requirements. A *FUN* slot would also be useful. If tasks G0001 and G0002 were identical, except that task G0001 was more fun, why not do G0001 first and let G0002 go until another day?

Now that a schema and some slots exist, let's add some values to the slots (Listing 2). These values will be used as defaults—that is, the values in the slots will be inherited into other frames if they're needed. Single or multiple values are added to a slot with *new-value* or *new-values*, which replace any previous values. Adding single or multiple values without destroying existing values is accomplished with *add-value* or *add-values*.

What do we have so far? We have defined a schema called *THING-TO-DO* and given it some slots and values. One of its slots is *FUN*, and we've given it a default value of *NOT-MUCH*. This schema doesn't have a lot of the slots a real *THING-TO-DO* would need (such as *DEPENDS-ON*, for example), but these can be added later as the need arises.

To see our *THING-TO-DO* schema printed on the terminal, another CRL function, *print-schema* is used. The schema is then displayed on the screen in print format. (LISP returns *T* for true after printing the schema):

```
> (PRINT-SCHEMA 'THING-TO-DO)
{{ THING-TO-DO
     PRIORITY: MODERATE
     RESOURCES: SYMBOLICS PRINTER
        CODING-PAD
     LOCATION: OFFICE
     DUE-DATE: 1-1-86
     FUN: NOT-MUCH}}
T
```

Just to be devious, let's try to add a value to a slot that does not exist (Listing 3).

Not a pretty sight! CRL comes back and tells us that it can't add a value to a slot if the slot doesn't exist. We could add the slot and then the value and everything would be fine. In CRL, the application developer can intercept these error reports and customize a message or recover from the error in any way he or she desires.

Let's create a schema called *PHONE-CALL* that can be used to represent the pertinent information in a telephone call. A LISP function called *CSCHEMA* (which is actually a macro) can be used to create a schema with its slots and values at one time rather than using the functions *CREATE-SCHEMA, CREATE-SLOT* and *NEW-VALUES*. *CSCHEMA* creates schemas such as *PHONE-CALL* with an arbitrary number of slots and values:

```
> (CSCHEMA 'PHONE-CALL
     ('IS-A 'THING-TO-DO)
     ('TOPIC)
     ('WITH-WHOM))
PHONE-CALL
```

To add a specific *PHONE-CALL* to the system, we again use the *CSCHEMA* macro. Since everything in CRL is a schema (Listing 4), a class such as *PHONE-CALL* and a specific instance of the class are both schemas. The slot *INSTANCE* tells the system that *PHONE-CALL-123* is a specific *PHONE-CALL*, which in turn is a *THING-TO-DO*. The relations between the schemas *THING-TO-DO, PHONE-CALL*, and *PHONE-CALL-123* are displayed in Figure 1.

If we want to find the value of a slot in a schema, the CRL functions *GET-VALUE* (to retrieve one value) or *GET-VALUES* (to retrieve more than one value) are used:

```
> (GET-VALUE 'PHONE-CALL 'IS-A)
THING-TO-DO
```

So we learn that a *PHONE-CALL* is a *THING-TO-DO*, which we knew already. Do we know anything about the location of *PHONE-CALL-123*, such as where I should make *PHONE-CALL-123* from?

```
> (GET-VALUE 'PHONE-CALL-123
    'LOCATION)
OFFICE
```

Although we hadn't explicitly entered a location to the *PHONE-CALL-123* schema, one can be assumed. The value of the location of *PHONE-CALL-123* is inherited from the schema *THING-TO-DO* via the *IS-A* relation. New relations can be added to the system readily, so any relation between schemas can be defined explicitly. CRL also provides features for including or excluding slots and values that can be inherited. *INCLUSION-SPEC* and *EXCLUSION-SPEC* schemas are used for this purpose.

Meta knowledge is knowledge about knowledge. Meta knowledge can be attached to a schema, slot, or value and used to record information about the knowledge base development, such as who added a particular schema and on what date. It can also be used to place restrictions on the values a slot can take and to set up demons or specify certain inheritance specifications.

Suppose I've already done my knowledge engineering task and I now know that valid PHONE-CALL topics must be related to a valid PROJECT, (another schema). I want the Things-to-Do Advisor to prevent somebody from coming in and saying that poetry is a valid phone call topic. This can be done by creating meta knowledge in the form of a restriction slot schema. A switch, which generally takes a *T* or *NIL* (true or false) value, must also be set to tell CRL to check restrictions:

```
> (CSCHEMA 'TOPIC
   ('RANGE '(SCHEMA (TYPE IS-A
      PROJECT))))
TOPIC
> (SET-SWITCH 'RESTRICT T)
T
```

Now if we try to set the value of the *TOPIC* slot of *PHONE-CALL-123* to *POETRY*, CRL will print an unpleasant "RANGE ERROR" message.

Once a domain of knowledge is properly represented in a network, the job of using the knowledge still remains.

```
> (CREATE-SLOT 'THING-DO-TO 'PRIORITY)
PRIORITY
> (CREATE-SLOT 'THING-DO-TO 'RESOURCES)
RESOURCES
> (CREATE-SLOT 'THING-DO-TO 'LOCATION)
LOCATION
> (CREATE-SLOT 'THING-DO-TO 'DUE-DATE)
DUE-DATE
> (CREATE-SLOT 'THING-DO-TO 'FUN)
FUN
```

Listing 1.

```
> (NEW-VALUE 'THING-TO-DO 'PRIORITY 'MODERATE)
MODERATE
> (NEW-VALUES 'THING-TO-DO 'RESOURCES '(LISP-MACHINE PRINTER CODING-PAD))
(LISP-MACHINE PRINTER CODING-PAD)
> (NEW-VALUE 'THING-TO-DO 'LOCATION 'OFFICE)
OFFICE
> (NEW-VALUE 'THING-TO-DO 'DUE-DATE '1-1-87)
1-1-87
> (NEW-VALUE 'THING-TO-DO 'FUN 'NOT-MUCH
NOT-MUCH
```

Listing 2.

We could use Common LISP exclusively. With this approach, the CRL schemas could be used as data structures only. The user could type in LISP expressions at the keyboard or write LISP programs that use CRL functions and schemas.

Another approach is to use demons. Here, schemas are attached to a slot, designated as demons and used to reference some code. When the slot is accessed (via *GET-VALUE*) the referenced code is called.

A third approach is to use object-oriented capabilities. In this scheme, functions are stored as values in slots. The functions are invoked when messages are sent to the schema containing the slot.

Inference capabilities are also provided with Knowledge Craft via PROLOG and OPS-5. These capabilities are not, however, part of CRL. In a real application, these control capabilities would be mixed and matched as appropriate.

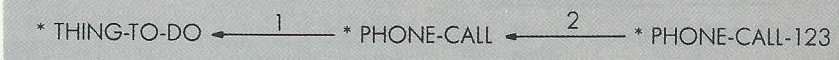
CRL also offers a mechanism for portraying alternative worlds in its efforts to deal with hypothetical or conflicting information. Suppose in the Things-to-Do Advisor I want to differentiate between several schedules in which the tasks can be executed:

```
> (CSCHEMA 'TASK-1 :CONTEXT 'ALL-
    GOES-WELL
  ('DURATION .5)
  ('PERSONNEL 1))
```

```
> (NEW-VALUE 'THING-TO-DO 'PROJECT 'EXOTIC-LANGUAGE-COLUMN)

***     CRL ERROR REPORT
***     TYPE                NO SLOT
***     SOURCE              NEW-VALUE
***     IN SCHEMA           THING-TO-DO
***     IN SLOT             PROJECT
***     WITH VALUE          NIL
***     IN CONTEXT          $ROOT-CONTEXT
***     AUXILIARY MESSAGE   "Slot doesn't exist"
```

Listing 3.

Schema relations



Relation keys:
1. IS-A
2. INSTANCE

Figure 1.

ALS Prolog™ 1.0

Real Prolog for Real Applications

- First true incremental interactive compiler for Edinburgh Prolog
  Needs no interpreter: compiles full Prolog "on the fly"
- Fast compiler execution and very fast code — LIPS on nrev:
  PC,XT: 3400 AT: 8 MHz: 14000 10 MHz: 17000
  VAXMate: 14000 Compaq 386: 31000
- Modules, complete garbage collection, tail recursion
- Standard debugger, assert, clause, etc. operate on compiled code
- C Interface, real strings, screen control, interrupt handling, DCGs
- Virtual code space
- Add on tools: Direct Dynamic Interfaces to dBase and R:Base
- Macintosh versions available shortly
- VISA, MasterCard, Drafts on U.S. Banks

Professional Version: $499
Personal Version: $199 also available
For more information or to order call Customer Service
(315) 471-3900

Applied Logic Systems, Inc. P.O. Box 90, University Station
Syracuse, NY 13210-0090 USA
315-471-3900

CIRCLE 49 ON READER SERVICE CARD

NGS FORTH

A FAST FORTH, OPTIMIZED FOR THE IBM PERSONAL COMPUTER AND MS-DOS COMPATIBLES.

STANDARD FEATURES INCLUDE:

- 79 STANDARD
- DIRECT I/O ACCESS
- FULL ACCESS TO MS-DOS FILES AND FUNCTIONS
- ENVIRONMENT SAVE & LOAD
- MULTI-SEGMENTED FOR LARGE APPLICATIONS
- EXTENDED ADDRESSING
- MEMORY ALLOCATION CONFIGURABLE ON-LINE
- AUTO LOAD SCREEN BOOT
- LINE & SCREEN EDITORS
- DECOMPILER AND DEBUGGING AIDS
- 8088 ASSEMBLER
- GRAPHICS & SOUND
- NGS ENHANCEMENTS
- DETAILED MANUAL
- INEXPENSIVE UPGRADES
- NGS USER NEWSLETTER

A COMPLETE FORTH DEVELOPMENT SYSTEM.

PRICES START AT $70

NEW◆HP-150 & HP-110 VERSIONS AVAILABLE

NEXT GENERATION SYSTEMS
P.O.BOX 2987
SANTA CLARA, CA. 95055
(408) 241-5909

CIRCLE 52 ON READER SERVICE CARD


```
TASK-1
> (CSCHEMA 'TASK-1 :CONTEXT 'ALL-
      GOES-POORLY
   ('DURATION 50)
   ('PERSONNEL 10))
TASK-2
```

When one context has been chosen as the correct version of the world (hopefully the *ALL-GOES-WELL* context!), the contexts can be merged. This means the schemas in the *ALL-GOES-POORLY* (or *ALL-GOES-WELL*) contexts will be removed.

CRL can be fine-tuned by using a number of switches that control the tasks CRL is supposed to do. These switches allow the user to adjust the environment to better suit it to the problem. Most of the switches are Boolean (taking *T* or *NIL* as values), but there are a few with three- or more-way settings. Examples include *CACHE* (whether to copy inherited values into schemas), *DEMON* (whether checks are made for demons), *RESTRICT* (whether to check for range restrictions), and *INVERSE* (whether inverse relations are automatically created), among some 20 others.

A very good example which demon-

```
> (CSCHEMA 'PHONE-CALL-123
           ('INSTANCE 'PHONE-CALL)
           ('TOPIC 'MEETING-FOR-LUNCH)
           ('WITH-WHOM 'REED)
           ('PRIORITY 'HIGH))
PHONE-CALL-123
```

Listing 4.

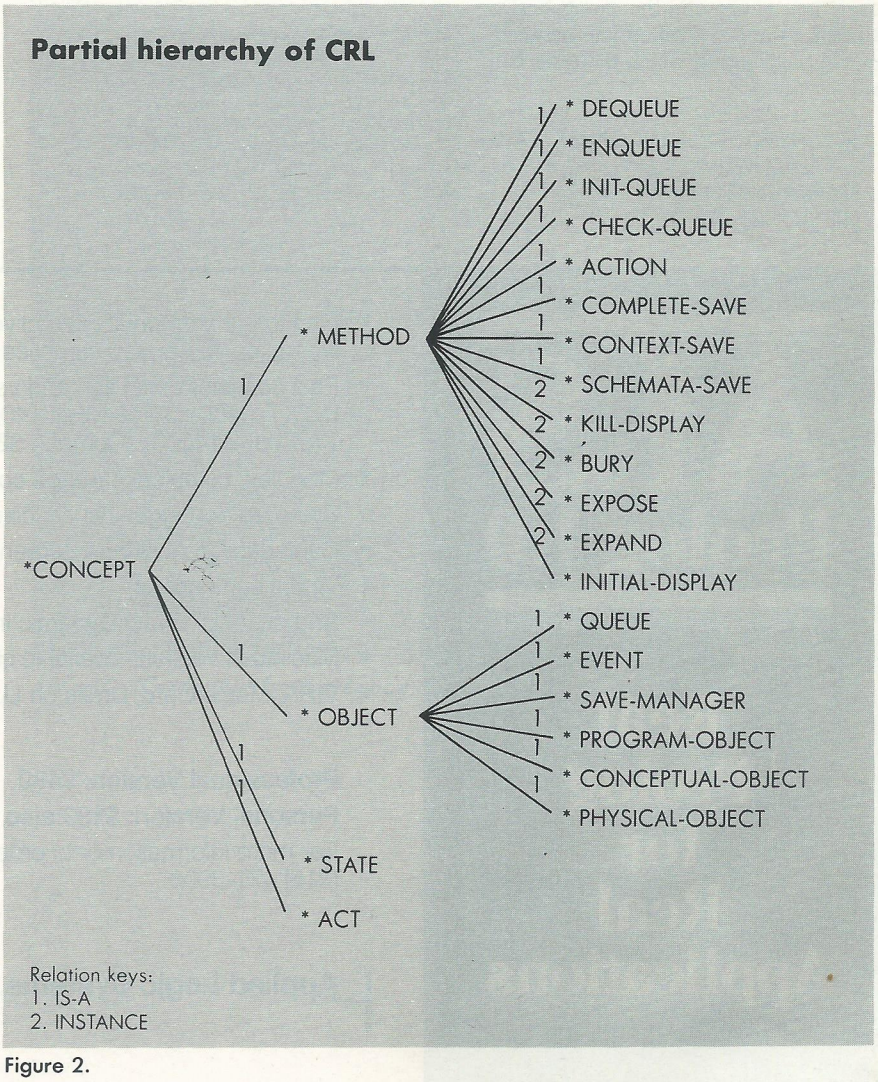


Figure 2.

strates the use of CRL to represent complex scenarios is found in a paper written by Arvind Sathi, Mark Fox, and Mark Greenberg,[7] in which schemas have been created for the important characteristics of project management, including abstraction, revisions, time, and constraints. The schema for *ELABORATION-OF* is shown in Listing 5.

This schema allows the *ELABORATION-OF* relation to be established between two schemas. This relation is similar to the concept of *ABSTRACTION,* where specific information is reduced along each link of this relation. *ELABORATION-OF* is a transitive relation, so if *A* is an *ELABORATION-OF B* and *B* is an *ELABORATION-OF C* then *A* is an *ELABORATION OF C.*

CRL understands the world through schema-colored spectacles. Everything of interest in CRL is a schema (frame). This includes relations, error recovery procedures, graphic objects, and utilities for manipulating CRL schemas.

In fact, the entire language can be represented in a hierarchy (partially pictured in Figure 2). This feature allows a CRL programmer to examine the CRL schemas (and possibly modify or extend them) as easily as the schemas that he or she has created. Listing 6 shows CRL schemas for *ERROR-SPEC, PROTOTYPE, EVENT,* and *DEMON.*

CRL offers many advantages to knowledge-based programming tasks. It provides an easy way to structure data so that it resembles the way we intuitively view it. Its uniform representation, in which everything is a schema that may contain slots and values, allows both computer manipulation and graphic representation.

Schemas can be used to capture both objects (such as *TASKS* and *PLANS*) and relations (such as *CONSTRAINED-BY* and *PROTOTYPE-OF*). In her book, *Artificial Intelligence,*[8] Elaine Rich states that to be useful, structured knowledge should possess four properties: representational adequacy, inferential adequacy, inferential efficiency, and acquisitional efficiency. In other words, the representation must be efficiently acquired, stored, and utilized.

CRL addresses these concerns to a large degree. The concepts behind CRL and the notion of knowledge representation can be useful for any programmer or analyst who must deal with the semantics of complicated knowledge.

References

1. Minsky, Marvin. "A Framework for Representing Knowledge." *The Psychology of Computer Vision,* P. Winston, ed. 1975.
2. Winograd, Terry. "Five Lectures on Artificial Intelligence." AIM-246. Stanford Artificial Intelligence Lab, 1974.
3. Kuipers, Benjamin. "A Frame for Frames: Representing Knowledge for Recognition." *Representation and Understanding,* D. Bobrow and A. Collins, eds. Academic Press, 1975.
4. Roberts, R., Bruce and Ira P. Goldstein. "The FRL Manual." AI Memo 409. Massachusetts Institute of Technology Artificial Intelligence Lab, 1977.
5. Lenat, Doug. "CYC: Using Common Sense Knowledge to Overcome Brittleness and Knowledge Acquisition Bottlenecks." *AI Magazine,* Winter 1985.
6. Wright, J.M., M.S. Fox, and D. Adam. *SRL/2 Users Manual.* Robotics Institute, Carnegie-Mellon University, Pittsburgh, Pa., 1984.
7. Sathi, Arvind, M. Fox, and M. Greenberg. "Representation of Activity Knowledge for Project Management." *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Sept. 1985.
8. Rich, Elaine. *Artificial Intelligence.* McGraw-Hill, 1983.
9. *Knowledge Craft 3.0 Reference Manual.* Carnegie Group Inc., 1985.
10. Bobrow, Daniel G. and Terry Winograd. "An Overview of KRL, a Knowledge Representation Language." *Cognitive Science* 1:1.

Doug Schuler is an AI specialist with the Advanced Technology Center of Boeing Computer Services. He is interested in knowledge representation and other issues underlying applications, such as computer-aided negotiation. He is currently coordinating the upcoming "Directions and Implications of Advanced Computing" Symposium sponsored by Professionals for Social Responsibility.


```
{{ ELABORATION-OF
    IS-A: RELATION
    DOMAIN: (OR (TYPE IS-A INDIVIDUAL)
                (TYPE IS-A PROTOTYPE))
    RANGE: (SCHEMA (OR (TYPE IS-A INDIVIDUAL)
                       (TYPE IS-A PROTOTYPE))
    INVERSE: HAS-ELABORATION
    INCLUSION: ELABORATION-OF-INCLUSION
    TRANSITIVITY: (REPEAT (STEP ELABORATION-OF T) Ø INF)
    COMMENT: "Elaboration-of defines abstraction"}}
```

Listing 5.

```
{{ ERROR-SPEC
   IS-A: PROGRAM-OBJECT
   TYPE:
   SIGNAL:
   ACTION:
   VALUE: }}

{{ PROTOTYPE
   IS-A: RELATION
   TRANSITIVITY: (STEP PROTOTYPE T)
   INVERSE: PROTOTYPE-OF }}

{{ EVENT
   IS-A: OBJECT
   QUEUE:
   EVENT-TIME:
   ACTION:
   NOTICE:
   DELETE:  }}

{{ DEMON
   IS-A: PROGRAM-OBJECT RELATION
   ACCESS:
   ACCESSOR
   ACCESS-VALUE:
   WHEN:
   ACTION:
   EFFECT:
   CURRENT-VALUE: }}
```

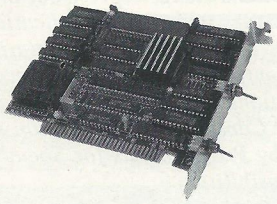
Listing 6.

MICROWAY ACCELERATES YOUR PC!

FastCACHE-286™

Runs your PC Faster than an AT!

Runs the 80286 at 9 or 12 MHz and the 80287 at 8, 9 or 12 MHz. Includes 8 kbytes of 55ns CACHE.



Compatible with Leading Edge Model D, Compaq, and Turbo motherboards. Includes 8088 Reboot Switch, DCache, Print Spooler and Diagnostics... **From $399**

LOTUS/INTEL EMS SPECIFICATION BOARDS

MegaPage™ The only EMS board which comes populated with two megabytes of cool-running, low power drain CMOS RAM installed. Includes RAM disk, print spooler, disk cache and EMS drivers. For the IBM PC, XT and compatibles...**$549**

MegaPage with ϕK..... **$149**

MegaPage with 2 megabytes of HMOS RAM..... **$419**

MegaPage AT/ECC™ EMS card for the PC AT and compatibles includes Error Correction Circuitry. With ECC, 11 RAM chips cover 256K so the user never encounters RAM errors. With 1 megabyte CMOS RAM..... **$699**

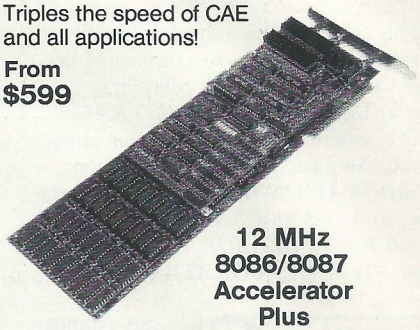
INTEL, JRAM, or Maynard ..... **CALL**

INTEL INBOARD 386 ϕK..... **$1325**

NUMBER SMASHER/ECM™

Triples the speed of CAE and all applications!

From $599



12 MHz 8086/8087 Accelerator Plus A Megabyte for DOS!

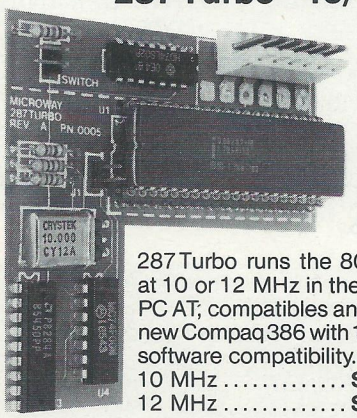
For the IBM PC, XT and compatibles

***PC Magazine* "Editor's Choice"**

8087 SOFTWARE

IBM BASIC COMPILER	$465
MICROSOFT QUICK BASIC	$79
87BASIC COMPILER PATCH	$150
87BASIC/INLINE	$200
IBM MACRO ASSEMBLER	$155
MS MACRO ASSEMBLER	$99
87MACRO/DEBUG	$199
MICROSOFT FORTRAN V4	$299
RM FORTRAN	$399
LAHEY FORTRAN F77L	$477
MS or LATTICE C	CALL
STSC APL★PLUS/PC	$450
STSC STATGRAPHICS	$675
SPSS/PC+	$695
87SFL Scientific Functions	$250
87FFT	$200
OBJ → ASM	$200
PHOENIX PRODUCTS	CALL

287 Turbo™-10/12



287 Turbo runs the 80287 at 10 or 12 MHz in the IBM PC AT, compatibles and the new Compaq 386 with 100% software compatibility.

10 MHz **$450**

12 MHz **$550**

***PC Magazine* "Editor's Choice"**

8087 UPGRADES

All MicroWay 8087s include a one year warranty, complete MicroWay Test Program and installation instructions.

8087 5 MHz..... **$105**
For the IBM PC, XT and compatibles

8087-2 8 MHz..... **$154**
For Wang, AT&T, DeskPro, NEC, Leading Edge

80287-3 5 MHz..... **$179**
For the IBM PC AT and 286 compatibles

80287-6 6 MHz..... **$229**
For 8 MHz AT and compatibles

80287-8 8 MHz..... **$259**
For the 8 MHz 80286 accelerator cards and Compaq 386

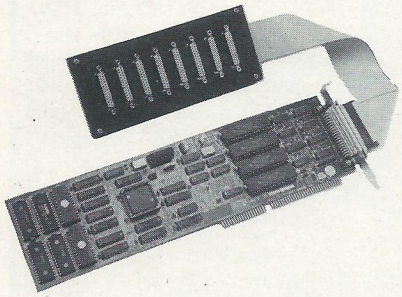
80287-10 10 MHz..... **$395**

PC-PAL™ Programmer..... **$395**

Call for great prices on V20, V30, 64K, 128K and 256K RAM

AT8™

Turns your AT into a high speed, multi-user Xenix business system!



8 port, intelligent serial controller with 3% response degradation. Includes 8 MHz 80186 with built in DMA **$1299**

MICROWAY SOFTWARE FOR LOTUS 1-2-3™

FASTBREAK™ employs the 8087 to increase the speed of Lotus 1-2-3™ Version 1A or 1A*. Users are reporting speed ups of between 3 and 36 to 1. When run with our NUMBER SMASHER accelerator card, recalculation speed ups of 10 to 30 are being reported..... **$79**

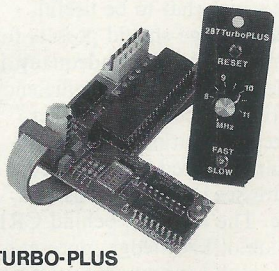
PowerDialer® Add-In for Lotus 1-2-3 Release 2. Automated telephone dialing from within 1-2-3. Adds least cost routing, automatic carrier selection and automated phone book worksheet. Builds customized dialing applications. Can be used with DesqView **$79**

HOTLINK™ adds easy linking of spreadsheets to Lotus 1-2-3 Version 1A... **$99**

287 TURBO-PLUS™

Speeds up your AT

Adjustable 80286 Clock 6-12 MHz
10 MHz 80287 Clock
Plus Full Hardware Reset..... **$149**
Optional 80286-10..... **$175**



287TURBO-PLUS
With 80287 10 MHz..... **$549**
With 80287 12 MHz..... **$629**

CALL (617) 746-7341 FOR OUR COMPLETE CATALOG

MicroWay
P.O. Box 79
Kingston, Mass.
02364 USA
(617) 746-7341

You Can Talk To Us!

MicroWay Europe
32 High Street
Kingston-Upon-Thames
Surrey England KT1 1HL
Telephone: 01-541-5466

CIRCLE 46 ON READER SERVICE CARD

PRODUCT BINGO

By Joanna M. Poole

Each month Product Bingo features the latest in new software and hardware products of interest to COMPUTER LANGUAGE *readers. Send new product information to Regina Starr Ridley, Editor,* COMPUTER LANGUAGE, *500 Howard St., San Francisco, Calif. 94105.*

Beginner? pro? C for yourself

Microsoft introduces two new versions of *Microsoft C: QuickC* for beginning C programmers ($99) and *Microsoft C Optimizing Compiler 5.0* for professional C programmers ($450).

Microsoft Corp., 16011 NE 36th Way, P.O. Box 97107, Redmond, Wash. 98073-9717, (206) 882-8080

CIRCLE 120 ON READER SERVICE CARD

C again

C:LINES/C:TREE prints C source code and produces a structural analysis of the code under PC-DOS.

23 Kent Pl. #3, Menlo Park, Calif. 94025, (415) 322-2679

CIRCLE 121 ON READER SERVICE CARD

FORTRAN forte

The latest release of *RM/FORTRAN, v. 2.4,* includes *RM/Forte,* a software development environment for the microcomputer. $595

Ryan McFarland, 609 Deep Valley Dr., Rolling Hills Estates, Calif. 90274, (213) 541-4828

CIRCLE 122 ON READER SERVICE CARD

Tools for text

QDIFF, QSRCH, and *QSE* comprise *TXTTOOLS,* a set of utilities providing programming and documentation aids for programming development environments. $85

Quilt Computing, 7048 Stratford Rd., Woodbury, Minn. 55125, (612) 739-4650

CIRCLE 123 ON READER SERVICE CARD

Numerical software in C

Find vector and matrix routines for scientific software developers in the *LINLIB* C library from *Information and Graphic Systems.* $150

Information and Graphic Systems, 15 Normandy Ct., Atlanta, Ga. 30324, (404) 231-9582

CIRCLE 124 ON READER SERVICE CARD

Solutions: software utilities

SoftLogic Solutions claims that its *Disk Optimizer* increases disk drive speed and its *Software Carousel* allows the loading of up to 10 programs at once. Each product is $59.95.

SoftLogic Solutions, 530 Chestnut St., Manchester, N.H. 03103, (800) 272-9900

CIRCLE 125 ON READER SERVICE CARD

Master your form

Version 3.0 of *SCREEN-ACE Form Master* for APL, C, and assembler is designed to create and maintain forms, tables, text screens, and menus on IBM PCs and compatibles. $195

ACE Software Products, Product Sales & Support, 6934 Petit Ave., Van Nuys, Calif. 91406, (818) 989-5329

CIRCLE 126 ON READER SERVICE CARD

In brief

The latest version of *Solution Systems'* general-purpose programming editor, *Brief 2.0,* includes an undo feature that allows programmers to reverse the effect of their last 300 commands. $195

Solution Systems, 335 Washington St., Norwell, Mass. 02061, (800) 821-2492

CIRCLE 127 ON READER SERVICE CARD

KEYSTONE unlocks an expert

KEYSTONE is an expert system development and delivery environment for the IBM PC/AT written in Golden Common LISP. $10,000 for site license.

Technology Applications Inc., 6621 Southpoint Dr. North, Ste. 310, Jacksonville, Fla. 32216, (904) 737-1685

CIRCLE 128 ON READER SERVICE CARD

Take advantage

Two new programming tools are available from *Lifeboat. ADVANTAGE Make* ($125) is an MS-DOS version of the UNIX make utility and *ADVANTAGE Disassembler* ($295) disassembles .EXE and .COM files.

Lifeboat Associates Inc., 55 S. Broadway, Tarrytown, N.Y. 10591, (914) 332-1875

CIRCLE 129 ON READER SERVICE CARD

Edinburgh PROLOG compiler

The *Cogent PROLOG Compiler* implements the Edinburgh standard dialect of PROLOG for IBM PCs and compatibles, offering over 150 predefined predicates. $200

Cogent Software Ltd., 21 William J. Heights, Framingham, Mass. 01701, (617) 875-6553

CIRCLE 130 ON READER SERVICE CARD

USE THE BRAINS YOUR IBM WASN'T BORN WITH.

Right at your fingertips in CompuServe's IBM® Forums.

Our IBM Forums involve thousands of users worldwide who will show you just how easy it is to get the most from your IBM and IBM compatibles.

The IBM New Users Forum lets you ask basic questions of PC experts. The IBM Junior Forum is perfect for PCjr® users. Trade tips with other IBM PC and AT users in the IBM Software Forum. Ask questions and get answers directly from the manufacturers in the PC Vendor Support Forum. And if you're looking for a PC Bulletin Board, visit the IBM Communications Forum. Or try the IBM Hardware Forum for discussions on hardware topics and product updates.

Easy access to free software, including free uploads.

You can easily download first-rate, non-commercial software and utility programs. Upload your own programs free of connect time charges. And take advantage of CompuServe's inexpensive weeknight and weekend rates, when forums are most active and standard online charges are just 10¢ a minute. You can go online in most areas with a local phone call. Plus, you'll receive a $25.00 Introductory Usage Credit when you purchase your CompuServe Subscription Kit.

Information you just can't find anywhere else.

Use the Forum Message Board to send and receive electronic messages. Join ongoing, real-time discussions in a Forum Conference. Communicate with industry experts, including the programmers who write your favorite programs. Search Forum Data Libraries for non-commercial software and shareware. Enjoy other useful services too, like electronic editions of popular computer magazines.

All you need is your IBM computer or IBM compatible computer (or almost any other personal computer) and a modem.

To buy your Subscription Kit, see your nearest computer dealer. Suggested retail price is $39.95. To receive our free brochure, or to order direct, call 800-848-8199 (in Ohio and Canada, call 614-457-0802). If you're already a CompuServe subscriber, type GO IBMNET (the IBM Users Network) at any ! prompt to see what you've been missing.

CompuServe®

Information Services, P.O. Box 20212
5000 Arlington Centre Blvd., Columbus, Ohio 43220

800-848-8199

In Ohio, call 614-457-0802

An H&R Block Company

CIRCLE 41 ON READER SERVICE CARD

Death by 1,000 *cuts*: PROLOG on the PC

By Steven Engle, Jonathan Hodgson, and James Vita

As personal computers have grown more powerful, they've become host to increasingly complex programs. Fifth generation languages once used solely on mainframes are now delivered on much smaller machines.

One such language is PROLOG. Unlike more conventional, procedural languages, PROLOG incorporates program control into the language itself. The programmer is relieved of the majority of the control of program flow and can focus on expressing data objects' relationships. Ultimately, this design vests the language interpreter with responsibility for most of a program's functionality, making the implementation of PROLOG on a microcomputer a formidable task.

As with other functional languages such as LISP and Forth, PROLOG applications are created by incrementally specializing the interpreter environment until the desired functionality is achieved. This method inevitably leads to problems with standardization and portability.

There is currently no universal PROLOG standard analogous to common LISP or FIG Forth. Over time, the implementation described by W.F. Clocksin and C.S. Mellish in *Programming in PROLOG* (Springer-Verlag, 1985) has emerged as the de facto standard. Few implementers are satisfied with merely following the standard; most choose to embellish their PROLOG environments with a variety of powerful and innovative capabilities.

The proliferation of nonstandard features is encouraged by a professional programming community generally more concerned with the requirements of the task at hand than virtuous adherence to a standard. Many implementations have successfully merged these two goals by offering a complete Clocksin and Mellish implementation and various extensions—in some cases, even alternative control structures.

As the language moves to the personal computer, implementers must come to terms with the machine's requirements and limitations. A PROLOG interpreter is memory intensive; data and programs must be available in memory. Consequently, one of the most pressing limitations is memory size.

The packages reviewed handle memory limitations in a variety of ways, ranging from virtual memory systems to compression schemes—techniques rarely visible to the user. Interestingly, none of the PC-based PROLOGs reviewed here makes use of the extended or expanded memory capabilities of the 80286 or 80386 processor, using instead methods that are compatible with standard PCs.

The packages also take several approaches to the delivery of applications. The most efficient approach is, of course, to compile interpreted code into a standalone, executable image. A disadvantage of this approach is the frequent need to introduce subtle changes into the interpreted code to facilitate compilation (such as typing of expressions and the declaration of predicates before they are used).

Other systems offer run-time environments that create essentially self-starting applications but otherwise continue to interpret code. A variation of this approach is systems that compile code into compact intermediate representations that execute faster and load more quickly.

In a similar vein, the integration of PROLOG applications with code in more conventional languages, such as C or Pascal, is interesting. Particularly intriguing are applications that make use of an existing data base product to extend the local PROLOG data base. The way this is done is by no means standardized; each product handles the interface differently.

Judging by the variety of products, the audience for PROLOG interpreters differs greatly from the users of more conventional languages. In many ways, the PROLOG market is still in its infancy, resulting in products that range from those produced by small groups of programmers in close contact with their user community to more polished efforts by larger organizations.

This diversity is healthy, and there's quality to be found over the entire spectrum of products. A PROLOG whose manual comes in a loose-leaf notebook is not necessarily inferior to one whose manuals are bound. The range of products available is indicative of the variety of PROLOG applications, spanning the complete range from academic research to professional product development.

The following products are covered: Applied Logic Systems' ALS PROLOG, Arity's Arity/PROLOG, Chalcecony Software's Chalcedony PROLOG/i, Cogent Software's Cogent PROLOG, Logicware's MPROLOG, Expert Systems International's PROLOG-2 Professional, Borland International's Turbo PROLOG, and Automata Design Associates' VML PROLOG. Tables 1 and 2 (pp. 132 and 134) summarize the benchmark results and features.

In general, the products reviewed were of consistently high quality. What separates them is the underlying engineering and the specific mixture of features that determine how well an interpreter will fit the application at hand.—S.E.

Applied Logic Systems
ALS PROLOG

Applied Logic Systems' ALS PROLOG comes on two disks: one for the system and the other with example programs, C interface files, and source code for many of the built-in predicates. Installation is completely straightforward and clearly explained at the beginning of the users manual.

ALS PROLOG permits access to DOS and an editor from within the program development environment. The package comes with two versions of Manx Software's Z text editor. One version is for true PC compatibles and directly manipulates the screen; the other uses DOS to address the screen. Z is a version of the Berkeley UNIX vi editor. If you don't want to use vi, it can be easily replaced with one you prefer. You can even change editors during an ALS PROLOG session.

ALS PROLOG is fairly close to the Edinburgh PROLOG of Clocksin and Mellish. The additional predicates Applied Logic Systems has included fall into three groups: the usual PROLOG extensions (*bagof, sort,* and so on), special PC extensions (DOS commands, a

command to invoke the editor, and BIOS services), and a special group of predicates related to implementation details of ALS PROLOG.

These last predicates are worth mentioning; they may help explain how ALS PROLOG achieves its amazing speed. ALS PROLOG is implemented using a threaded code interpreter for an abstract PROLOG machine. A manual section gives some details on how you can access the machine's code generator through the built-in predicate *$icode*.

Interestingly, even predicates that undergo assertion or retraction are compiled by the compiler. In addition, the compiler does complete garbage collection on both *heap* and *trail*. The compiler implements last-call optimization—predicates that use tail recursion are optimized as a special case.

Using the system is quite straightforward. To consult files, put their names on the command line when you invoke the compiler; they're automatically reconsulted when you return from the editor. It's worth noting that if a file name includes the .PRO extension when it appears on the command line, the compiled file won't be written to disk. If the file doesn't include the .PRO extension, the name will be given a .OBP extension and the compiled file written to disk. As a result, files invoked with the .PRO extension run slightly slower than those in which the compiled version was written back to disk.

ALS PROLOG does not produce stand-alone code. However, if you put a suitable goal in the files that are consulted on the command line, you can produce a program that can be run in a batch file. (Returning to DOS requires the use of the undocumented predicate *halt*.)

The example programs on the second disk cover a wide variety of problems and are well worth studying. In some cases, the source code has better documentation than the manual. There needs to be a more complete explanation of invoking programs. One particularly impressive example is a programmable desk calculator that makes use of definite clause grammars, useful for writing parsers. (See the accompanying sidebar "PROLOG benchmarks," pp. 130-131.)

While most of the error messages were helpful, some were less so. For example, if you're on the command line and try to consult a file that's not on the drive specified, the error message will be "error loading file <filename>." Unfortunately, this message covers many possible errors, including the destruction of a file. While this is an extreme example and most of the other error messages are better, I don't think that just because a compiler is directed to professionals you should skimp on things like error messages.

The manual is on the terse side and definitely written for experts. To be fair, it does come with an order form for Ivan Bratko's *PROLOG Programming for Artificial Intelligence* (Addison-Wesley, 1986).

I have a couple of minor complaints. Since some of them are about things that can affect your programming, I'll list them.

ALS PROLOG does not mind if the right-hand side of the predicate *X is Y* is uninstantiated, preferring to fail the goal at this point. This certainly does not accord with Clocksin and Mellish, although it could be used to initiate backtracking in some situations.

ALS PROLOG normally only produces the first instantiation of the variables in a goal. After writing them to the screen, it stops on the same line as the output. Nowhere in the manual does it tell the user that a semicolon will cause ALS PROLOG to try and redo the goal. I mention this because although this feature is fairly standard in PROLOG implementations, it is by no means universal.

On a more positive note, ALS PROLOG provides support for creating large programs through the use of modules. They're similar to the modules in Modula-2 in that predicates must be explicitly exported from a module if they are to be used elsewhere. Since you can't ac-

FROM THE DEVELOPERS OF JYACC FORMAKER

JYACC

APPLICATION MANAGER

A COMPLETE AUTHORING SYSTEM FOR SOPHISTICATED APPLICATIONS

FOR DEVELOPERS IN UNIX™, XENIX™, MS-DOS™ and PRIMOS™

Superior because the JYACC APPLICATION MANAGER Combines the display authoring capabilities of JYACC FORMAKER with facilities for linking display screens, data entry forms, menus and application programs.

- PROTOTYPE NEW INTERACTIVE APPLICATIONS
- IMPLEMENT TRANSACTION APPLICATIONS
- ENHANCE EXISTING APPLICATIONS WITH MENUS, INSTRUCTION DISPLAYS AND HELP WINDOWS
- CREATE CUSTOM APPLICATION AND DEVELOPMENT ENVIRONMENTS
- PRODUCE INTERACTIVE TUTORIALS

JYACC, INC. is a project-oriented computer consulting firm providing services in most areas of system design and implementation. Call us today to discuss your specific needs and applications.

JYACC INC.
116 John Street
New York, New York 10038
212-267-7722
Outside New York call 1-800-458-3313

UNIX is a trademark of AT&T, XENIX and MS-DOS are trademarks of Microsoft, PRIMOS is a trademark of Prime Computer

CIRCLE 55 ON READER SERVICE CARD

cess predicates that haven't been exported from a module, information hiding is possible. However, information hiding can't be completely realized since you can still explicitly assert into or retract from a module.

When a module is loaded, invocation of the predicate listing will display all its predicates on screen, including those that are not exported. In spite of these quirks, ALS PROLOG's support of modules is indispensable for large programs.

The usual debugging facilities provide for tracing a goal. In addition, spy points can be set and leashing is permitted. PROLOG debuggers inevitably produce large amounts of output, and this one is no exception. If you do invoke any of the debugging predicates, the debugger .OBP file is consulted. Otherwise it is not normally loaded, saving room in the symbol table.

How well does ALS PROLOG perform as a compiled version of PROLOG? I've mentioned that it is amazingly fast. On a Zenith AT-compatible that's about twice as fast as IBM's PC/AT, programs ran at a consistent 18,000 logical instructions per second (LIPS). On an Eagle PC-compatible, performance was 3,000 LIPS.

While the benchmark suite required minimal adaptation, some of them could not be modified without losing much of their significance. ALS PROLOG's compiler restricts the number of arguments to a predicate to 15 or less in accordance with the 16-register architecture of the compiler's abstract target machine. One consequence of this is that the call *functor(T,f,16)* will fail.

I ran most of the examples that came with the system and found that they all worked as advertised. As mentioned, the examples are of high quality and reward careful study.

Since I didn't have any large programs available, I couldn't test the compiler's support for procedure swapping. This facility keeps track of the least frequently used procedures and swaps them out of memory when more space is needed.

Any programming environment should provide some hooks into the operating system. ALS PROLOG offers two methods: built-in predicates to do the work of C library functions of the same name (essentially, it forks a process), and other predicates to address video and keyboard BIOS routines. These are supported by SCREENS.PRO, a module that provides various graphics primitives. If you have Manx's Aztec C compiler, you can also call C programs from ALS PROLOG, but this is the only compiler presently supported. It seems a pity that the DOS function calls are not directly supported.

For a software development environment, ALS PROLOG's restriction that file handling be done only through *see* and *tell* is a severe limitation that makes accessing data base files difficult. However, Applied Logic Systems has announced a separate package that will interface ALS PROLOG to several of the standard data base packages, which should considerably enhance the value of ALS PROLOG for applications development.

I found ALS PROLOG a very comfortable environment to work in. I appreciate the close adherence to the Edinburgh standard and the truly amazing speed of execution. ALS PROLOG is a complete package suitable for many AI applications, particularly systems run from some top-level interface or those that will add their own rules. The limitation on the number of arguments to predicates could hinder data base applications since it will restrict the number of attributes a relation can have.

Although ALS PROLOG does not offer the kind of bells and whistles a windowed environment with pop-up menus does, it does present a smooth interleaving of editing and testing, very much in the spirit of UNIX. Programmers who like the UNIX environment will probably like this compiler. In spite of some reservations about the documentation, I really liked the product. Anyone interested in writing large programs in PROLOG should check it out.—J.H.

Arity Corp.
Arity/PROLOG
Arity Corp.'s Arity/PROLOG is an implementation of the Clocksin and Mellish

JYACC FORMAKER

A POWERFUL SCREEN AND WINDOW MANAGER

UNIX™ ■ XENIX™ ■ MS-DOS™ ■ PRIMOS™ ■

JYACC'S FORMAKER makes it easy to design, develop, test and document interactive applications. **FORMAKER** includes a utility for creating and maintaining forms and windows and a subroutine library to provide access to them.

- Reduced Development Time
- Less Complex Programs
- Advanced User Interface
- Easy Form Creation
- System Prototyping
- Self-Documenting
- Application Portability

- Display Forms
- Windows Management
- Edits and Validations
- Display Prompts
- Error Messages
- Save Data
- Restore Data
- Cursor Control

- Easy-To-Use
- Free-Form Design
- "Test Mode"
- Pop-Up Windows
- Interactive Form Editing

JYACC, INC. is a project-oriented computer consulting firm providing services in most areas of system design and implementation. Call us today to discuss your specific needs and applications.

Available for the IBM PC/XT/AT and compatibles, AT&T 7300 and 3B family, DEC Vax and Micro VAX, NCR Tower, Prime 50 series, Altos 986, Fortune 32:16, Gould Concept series, HP 9000 and TI PC family.

JYACC INC.
116 John Street
New York, New York 10038
212-267-7722
Outside NY call 1-800-458-3313

UNIX is a trademark of AT&T
XENIX and MS-DOS are trademarks of Microsoft
PRIMOS is a trademark of Prime Computer

CIRCLE 56 ON READER SERVICE CARD

standard for professional software development. The system I reviewed included an interpreter, compiler, and sets of development utilities and example programs. The package provides extensive support for a wide set of commonly used PROLOG extensions, including second-order predicates, alternative control structures, and data base control predicates. Also included is support for definite clause grammars and extensive access to PC BIOS functions.

Arity/PROLOG has an introductory manual, installation guide, reference manual, and compiler guide and is delivered on four noncopy-protected disks. To install the program, simply copy the floppies to the hard disk and define two DOS logical names to identify the editor and the path to the nonexecutable system files. Instructions are included for installation to floppy-based systems, though the usefulness of such a system is questionable. The package does not include an editor, though most any file editor capable of working with straight ASCII files is accessible without leaving the interpreter.

While the documentation is thorough and well executed, it is directed at the professional user and does not introduce PROLOG to beginners. The installation guide provides a quick set of examples on how to work with the interpreter, clearly explaining starting the interpreter, exiting to DOS, and other tasks that do not fit well into the language manual.

The language manual is complete, easily navigated, and includes an index (eliminating linear searches of the manual). Each predicate is presented with examples; the more involved sections include several introductory pages. The absence of the obligatory 10-page introduction to PROLOG signals the package's intended audience.

The third manual details the operation of the compiler and procedures for building stand-alone, compiled applications. These applications do not require an interpreter, making Arity/PROLOG a candidate for developing marketable products. This manual is detailed, well structured, and, again, clearly intended for the experienced user.

Arity fully implements the Clocksin and Mellish standard, including definite clause grammars and second-order predicates like *bagof, findall,* and *setof.* Despite adherence to the standard, usability has not been sacrificed. The standard functionality is extended in a variety of ways, resulting in a complete system that is more usable.

Arity includes a special *counter* data type and a number of operators for bitwise operations and ordering and sorting atoms. In many cases, both integer and floating-point versions of the generic arithmetic operators are provided for use when efficiency is a concern.

Arity departs from the Clocksin and Mellish standard with its handling of the string data type. Rather than store strings as lists of atoms or lists of characters—both awkward for programmers and users—Arity stores strings as arrays of integer values delimited by the dollar sign. This approach conserves memory space while still allowing keyword searches, natural user input, and string manipulation. Built-in predicates support manipulation and conversion of strings to various other data types.

Arity provides two interesting, nonstandard control structures: *case* and *snip.* The *case* operator alters the goal according to the value of an expression. Its syntax is illustrated in the example from the Arity manual shown in Listing 1. Note the vertical line between the final two terms that delimits the default case.

The *snip* operator extends PROLOG's venerable *cut* operator (*!*), providing more control over backtracking. *Snip* is essentially a *cut* whose scope can be limited to a set of contiguous terms. In the following case, the failure of the predicate *E* would result in a failure of the entire expression: *A :- B, C, D, !, E.*

In contrast, if you use *snip* in the following expression, the failure of *E* causes backtracking to proceed from the *B* term: *A :- B, [!C, D!], E. Snip* is used to limit the search space but not as drastically as a *cut* would.

Arity extends the PROLOG's data base capabilities with three methods for organizing data bases: worlds, B-trees, and hash tables.

Worlds partition a data base, thus reducing the size of a search space. Worlds can be categorized as code worlds, which contain clauses and predicates, and data worlds, which are limited to data. However, unless otherwise specified, all actions occur in the default world, thus maintaining compatibility with applications that don't make use of this feature. Worlds can also be saved to files, allowing limited overlaying and freeing data-intensive applications from the memory constraints of a PC host.

Balanced binary trees (B-trees) and hash tables are accessible through a series of predicates similar to the standard data base access predicates. The predicates accept a tree or hash table name, a key, and the term to be stored in the structure. Because Arity provides access to the absolute data base location of any given predicate, it is also possible to store indexes in B-trees or hash tables, allowing you to index the data base under multiple keys.

The package provides the full complement of I/O predicates as defined by Clocksin and Mellish. Several additions are made to support IBM PC-specific functions, such as retrieving keyboard scan codes, accessing directories, and forcing duplicate file handles. Random access to files is supported with a predicate for specifying an offset address relative to the beginning of a file. Low-level access to PC ports is also provided.

Arity allows access to DOS functions through a variety of built-in predicates, including text-screen management with scrolling subregions of the screen, file management, and date functions. Windows are not directly supported, though implementing them should be within most users' grasp.

One glaring omission is the lack of access to the interrupt instruction. Since this could be added to the interpreter in a variety of ways, it's odd that it's not included, forcing programmers to devise their own means of accessing functions that are not built-in.

True to its stance as a serious development system, Arity/PROLOG includes a complete set of debugging tools. Full support is provided for the procedure-box model, allowing traces to be associated with all four ports of the standard execution model. The volume of trace information can be controlled using the leash predicate, reducing superfluous trace information.

Arity/PROLOG is not intended for the casual user. The manual details how to interface the language with a wide variety of languages. The package provides extensive compilation control, including generating *FAR* calls, declaring data types for code optimization, and limiting the accessibility of particular predicates. The package includes a family of predi-

```
case([ X = 0 -> write($The day is Sunday$),
       X = 1 -> write($The day is Monday$),
       X = 2 -> write($The day is Tuesday$),
       X = 3 -> write($The day is Wednesday$),
       X = 4 -> write($The day is Thursday$),
       X = 5 -> write($The day is Friday$),
       X = 6 -> write($The day is Saturday$) |
                write($Invalid date$)])
```

Listing 1.

C BRICKLIN RUN

Data Entry • Menus • Windows • Prototyping • Database • Toolkit

C-scape

■ Total Screen Control/Easy to Use

C-scape is a combination screen generator and library of screen I/O functions. Written for C programmers, C-scape brings a proven approach to the need for an easy-to-learn and use, but truly powerful and flexible screen management tool.

C-scape's kernel is your most powerful ally. Without requiring parameters you'll never use, it allows you to create tailored functions with ease and simplicity. Each key is individually definable. If you know *printf*(), you can use C-scape. C-scape's kernel provides a veritable screen design and construction toolkit to rewrite our functions or to write your own.

■ Most Powerful Prototyping Available

C-scape offers a unique approach to prototyping your software. You may use **Dan Bricklin's Demo Program** to create, edit, and view your screens (you can even capture existing screens from other programs), and then use C-scapes's **demo2c** utility to convert each screen to code.

You can design each screen with attributes such as colors, menu selections, data entry fields (including type, validation, and field naming), masking, and text, and then automatically convert the entire screen to code.

■ Powerful Function Library

Use C-scape's functions for Lotus-like, pull-down, or your own menu designs, automatic scrolling, pop-up windows (number limited only by RAM), logical colors, help, time and date, yes/no, tickertape fields, secure and protected fields, and many others, to turn your demo into a fully functioning and complete program in a fraction of the time spent coding screens from scratch.

C-scape's extensive library includes just about all the data entry and display functions you'll ever need, including money functions, fully definable borders, and orthogonal field movement (get the latest list by calling for more information). And modifying our functions or writing your own is easy. C-scape adjusts automatically for CGA, EGA, monochrome, and the Hercules Graphics Card Plus in RamFont mode, and optionally writes directly to video memory, so it's flexible *and* fast.

■ Bridges to Power

C-scape includes examples of how to bridge to other powerful tools such as **c-tree** and **db_VISTA.** You'll be integrating demos to dictionaries to file handlers and database managers in no time. You can even use C-scape to provide the screen design for AI applications, using packages that support calls to C.

■ Clean, Complete Documentation

C-scape's documentation is a clear example of how to write for programmers in a hurry. A short introduction uses helpful examples to explain the C-scape design. Each function is documented separately. An index makes reference easy, and a quick-reference card provides a synopsis of each function.

■ Source Code Included/Portable/No Royalties/No Runtime License

Providing source code at no additional cost gives you the freedom to modify existing functions without raising cost as a barrier. The source code includes all the low level routines you might need to port C-scape to an unsupported machine or compiler. Speaking of barriers, you pay *no royalties or runtime license fees,* either.

■ Toll Free Support and Bulletin Board

To make your job even easier, we provide technical support (toll free), and access to our 24-hour Bulletin Board.

■ Easy Prices/Risk-Free Terms

Try C-scape for 30 days. If you are not satisfied, return it for a refund. When you register, we send you source code. Order C-scape today, or call toll free for more information. See why some very major companies are standardizing on C-scape.

C-scape (Lattice/Microsoft/others)	**Only $199**
C-scape with Dan Bricklin's Demo Program	**$259**
NEW **C-scape with DB Demo and db_VISTA** (RAIMA)	**$425**

Please add $3 for first class shipping; Massachusetts orders add 5% sales tax.

Oakland Group, Inc.
675 Massachusetts Avenue, Cambridge, MA 02139-3309

MasterCard VISA **CALL NOW!**
800-233-3733
617-491-7311

CIRCLE 60 ON READER SERVICE CARD

cates for interfacing to assembler, Microsoft and Lattice C, and Microsoft FORTRAN and Pascal modules. The compiler package also includes a lint facility for locating potential trouble spots prior to compilation.

Assembly language routines can be added to the interpreter environment's built-in predicates with a declaration compiler. To do this, compiled predicates are added to the code world of the interpreter, and matching PROLOG predicates are declared to access the new predicate.

Arity/PROLOG offers a powerful and well-appointed development environment. It's a professional-class system suitable for both product development and research. The system is well documented and supported. The package has the style of a faithful implementation of Clocksin and Mellish PROLOG while meeting the needs of the software professional. —S.E.

Chalcedony Software
Chalcedony PROLOG/i

Chalcedony (pronounced "kal-SED-ony") Software's PROLOG/i is a significant advance over its earlier PROLOG/V. While still an interpreter, PROLOG/i supports large-memory-model applications (up to 640K RAM), math coprocessors, user-written overlays, and operating system commands, as well as a rich screen- and file-handling vocabulary.

Chalcedony's PROLOG/V and PROLOG/V Plus have been on the market since about 1982. As their successor, PROLOG/i retains compatibility with those products, as well as adhering to the Clocksin and Mellish standard.

PROLOG/i's user interface resembles a PROLOG user dialogue, similar to that described in Clocksin and Mellish. However, the dialogue differs somewhat from most other implementations' and the Clocksin and Mellish standard. If you enter a clause followed by a period at the user prompt (>), the clause is assumed to be a term and automatically asserted into the data base. To query the data base, a term must be followed by a question mark.

I can't fault the choice of a period for an assertion and a question mark for a query, as these are certainly logical assignments. However, this deviation is somewhat distracting to the flow of the PROLOG dialogue. I often want to consult files and query facts in these files. On most keyboards, the PROLOG/i syntax requires an extra, shifted key press. The deviation is less convenient than the more familiar syntax in which a period represents a query. However, this quirk is easily learned and becomes less distracting with practice.

PROLOG/i comes with its own editor so you can immediately begin building real PROLOG files. The editor somewhat resembles WordStar, but the command sequences are totally different. Many users will not be interested in learning the syntax of yet another editor. For them, detailed instructions are given for installing their own word processor or text editor that can be called from the PROLOG/i environment.

One major addition to the standard Clocksin and Mellish predicates is the ability to save the current data base. The *save* predicate dumps the current data base (minus comments) into a file, a feature particularly useful for experimentation.

The ordinary predicates *consult* and *reconsult* can also be used to read a data base back into the PROLOG/i work area. However, the additional predicates *load, reload,* and *unload* can be used to achieve effects not attainable with standard PROLOG. When *load* reads clauses from a file, PROLOG/i keeps track of the files they came from. The predicate *reload* acts like *reconsult,* affecting only the clauses associated with a particular file.

The built-in predicate *unload* selectively removes any clauses that were loaded from a specified file (acting like an all-encompassing, though selective, *retract*). Taken together, these features give the programmer total control over the data base and can be invaluable during a debugging session. PROLOG/i does not support the *[user]* syntax for adding terms to the data base from the keyboard. Instead, the user can consult the special file *con:*.

PROLOG/i provides a rich superset of the features of standard PROLOG. The *op* predicate is supported, allowing the user to define new operators. PROLOG/i handles PROLOG terms as data objects using the familiar *functor, clause, term, arity,* and *arg*. PROLOG/i does not support the dot (.) as either an infix or prefix operator on a list. The *op* predicate cannot be used to define such an operator. Nevertheless, a powerful set of list-handling features are provided. Multiple declarations are supported (predicates with the same functor name but differing in arity).

The standard arithmetic and comparison operators are included with only minor variations (for example, a normal slash instead of a backslash for /==). PROLOG/i handles floating-point numbers, as well as signed integers in arithmetic expressions.

Special formatting predicates allow the precision of results to be controlled. For example, floating-point output can be specified to be in scientific notation or fixed-point notation with a specified number of significant digits before and after the decimal point. In addition to the standard arithmetic predicates, PROLOG/i provides a full range of extended functions including double-precision arithmetic, exponentiation, and trigonometric and transcendental functions.

File I/O is handled with the standard *see, seeing, seen, tell, telling,* and *told* predicates. In addition, PC/MS-DOS users can manipulate directories and default drives with *chdir* and *chdisk*. Output can be directed to the screen or printer. For owners of computers that aren't completely IBM-compatible, ANSI screen I/O is possible once the ANSI.SYS device driver (not provided) is properly installed.

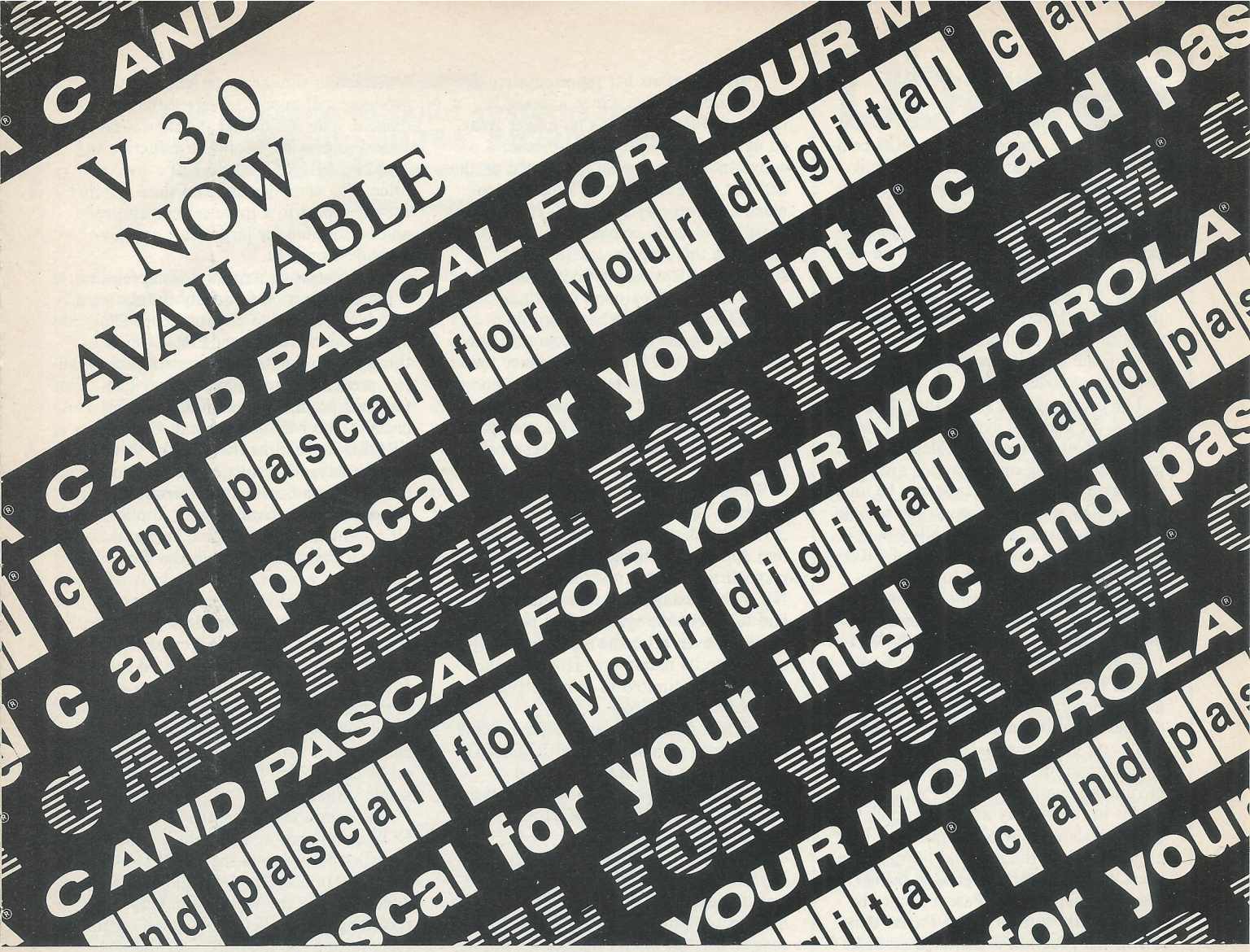
Users with color graphics (CGA) systems can take advantage of some machine-dependent graphics features. Predicates are provided to set the display mode of the computer and to read and set individual pixels on the display with up to two color palettes. PROLOG/i supports all of the standard IBM display modes, including high resolution (640 x 200 pixels).

With these features, you can make some interesting forays into the world of pattern generation and recognition. Limited text-oriented graphics are also supported. In that mode, you can do such things as clear the screen, plot the cursor, and control the scrolling of windowed screen areas.

PROLOG/i's debugging facilities are essentially identical to those described in *Programming in PROLOG*. You can leash the various PROLOG ports (*CALL, EXIT, FAIL,* and *REDO*) for a step-by-step trace of a program's execution. To debug new code, individual predicates can be designated as spy points; tracing can be turned on or off at any time. Debugging is performed in the main window as part of the normal PROLOG/i dialogue.

The PROLOG/i package has two manuals. The first manual, essential to the new user, is a language reference and tutorial. Its 184 pages include a brief introduction to PROLOG techniques, an indexed listing of built-in predicates, a listing of PROLOG dialogue error messages, and a glossary. This manual is not intended to be a complete course on PROLOG; for further information, the user is referred to *Programming in PROLOG*. The second manual is a users guide, a detailed, 76-page treatise on the editor and advanced topics.

PROLOG/i can be extended by manipulating the overlay library where many of the built-in predicates reside. Most of the PROLOG predicates are part of the system kernel. However, PROLOG/i maintains a transient overlay area that it uses to run DOS commands and call



Whitesmiths, Ltd. Has The Compiler You Want On The Machine You Use.

For over eight years Whitesmiths has focused its efforts solely on developing and supporting a family of quality systems software. Today, Whitesmiths is the only company offering compatible C and Pascal native and cross compilers for the full spectrum of computers on the market—from the IBM PC to the IBM 370, from the DEC Micro-11 to the VAX 8600, and all of the most popular processors in between.

Version 3.0 of Whitesmiths compilers is now available, with new features and functionality designed to meet the needs of today's professional software developers. Features include a C source level interactive debugger with breakpointing and variable display, a powerful multi-segment linker for generating ROM-based code, and the ability to display high-level source code, assembly language, and machine object code on one listing.

With Version 3.0, you get features supporting the emerging ANSI C standard, plus a uniform run-time environment and identical source code across all machine architectures.

The Version 3.0 Pascal compiler provides you with a Pascal to C Translator, a full ISO Level 1 implementation that includes conformant array parameters and numerous extensions.

If you're in the market for C or Pascal, call Whitesmiths at **1-800-225-1030.** We have the compiler you want for your machine.

Whitesmiths, Ltd.

59 Power Road, Westford, MA 01886 • (617) 692-7800 / Telex 750246

INTERNATIONAL DISTRIBUTORS: **AUSTRALIA**, Whitesmiths Australia, P.O. Box 21, 51 Grantham Street, Carlton, NSW 2218, (612) 588-7652 • **FRANCE**, COSMIC S.A.R.L., 52 Quai des Carrieres, 94220 Charenton Le Pont, Paris, (14) 378-8357 • **GERMANY**, GEI, Gesellschaft fuer Elektronische, Informationsverarbeitung MBH, Pascalstrasse 14, D-5100 Aachen, 02408/13-0 • **JAPAN**, Advanced Data Controls Corp., Nihon Seimei Otsuka Bldg., #13-4, Kita Otsuka 1-Chome, Toshima-ku, Tokyo 170, (03) 576-5351 • **SWEDEN**, Unisoft AB, Fiskhamnsgatan 10, S-41455 Goteborg, (31) 125810 • **SWITZERLAND**, RETIS, Realtime Software AG, CH-5001 Aarau, Bahnhofstrasse 96, (64) 247777 • **UNITED KINGDOM**, Real Time Systems Ltd., P.O. Box 70, Douglas, Isle of Man, (624) 26021.

CIRCLE 69 ON READER SERVICE CARD

predicates that reside in external libraries.

Several sample programs demonstrate how compiled, Pascal, C, and DOS command programs can be executed from within a PROLOG dialogue or program using the *loadexec* predicate. In addition, several examples illustrate how the external PROLOG/i library can be extended by including compiled C programs. (The technique involves relinking the existing library with the compiled versions of the predicates.) To use this system, the compiler must produce standard object-code output.

The users manual states that the format of the external overlay area is proprietary. Consequently, the details of performing these modifications are somewhat sketchy, although they can be understood by working through the sample source programs. This capability is not essential to the basic operation of PROLOG/i and is better left to the experienced DOS programmer.

PROLOG/i comes on two noncopy-protected disks. The first has the PROLOG/i interpreter, library, and library linker; the second, a few sample programs. Several of the sample programs are described in the language reference and users guide manuals.

The program disk includes examples of running compiled C and Pascal programs in the overlay area. A sample C program shows how to compile and link a program into the external library. The users guide makes reference to using these techniques with Pascal programs but explicitly states that no Pascal demonstration program is included on disk.

The sparsity of sample programs is the only shortcoming of PROLOG/i's reference materials. Overall, the documentation is accurate and well written.

Technical support is not included in the purchase price. Instead, the user is encouraged to join a users group at a modest cost. Joining the group entitles you to a quarterly newsletter and access to a resident programmer. Owners of PROLOG V and PROLOG V Plus can upgrade to PROLOG/i for $20. Version updates are available to registered users for the cost of shipping and handling.

PROLOG/i generally performs as expected. One inconsistency is its treatment of strings. Standard syntax treats a string as a list of ASCII characters (literally, a list of numbers that represent the ASCII characters). PROLOG/i treats strings as a list of characters represented as atoms (the characters themselves). Several predicates (notably, those that allow external program calls) require strings as arguments.

A string and its list representation are not interchangeable. Once a string is read into PROLOG, it's no longer treated as a string. A predicate to build a command string can be performed on the list, but these lists are no longer recognized as strings by PROLOG/i. The list must be formally converted back into a string. In this respect, treatment of strings has a strongly typed nature. Apart from its treatment of strings, PROLOG/i is otherwise untyped.

Assessing the speed of programs run under PROLOG/i is difficult since the overlay library only provides the crudest of mechanisms for timing execution speeds. However, PROLOG source programs load and execute with reasonable speed.

When reading one of the benchmark files that had approximately 150 tail-recursive predicates, the PROLOG/i system appeared to hang and did not respond to a keyboard-executed break. No error message was reported, and I had to reboot the system. The problem persisted even after the stack size was increased to its maximum. However, the problem was limited to this one type of PROLOG source file.

I should note that this excessive use of tail recursion is neither good PROLOG programming practice nor typical of most PROLOG programs. With well-designed programs, PROLOG/i performed admirably.

For both novice and veteran, Chalcedony Software's PROLOG/i will be a rewarding experience. PROLOG/i is very much a programmer's PROLOG. It's easy to use but allows further growth as experience is gained and expertise developed. Its expandability, graphics, and high-level arithmetic support make it a worthy candidate for large-scale AI systems development.—J.V.

Cogent Software
Cogent PROLOG

Most microcomputer language products strive to achieve a certain syntactic richness while maintaining compatibility with a de facto standard. Living up to its name, Cogent PROLOG quietly accomplishes both goals.

The veteran PROLOG programmer with knowledge of the standard PROLOG syntax will find Cogent an easy way to transfer that experience to the micro. In its literature, Cogent Software describes its PROLOGs as being very close to the Clocksin and Mellish standard. With only a few exceptions, Cogent PROLOG lives up to this claim.

The Cogent PROLOG interpreter appeared about one year ago and is currently in its third version. As of my writing this review, the compiler and runtime system were scheduled for release in early May 1987.

Although similar in design, the two products will appeal to very different audiences. The interpreter package is designed to provide a solid introduction to standard PROLOG syntax at a modest price. The compiler and run-time system should appeal to a more serious audience and be suitable for large-scale applications development.

The interpreter's most striking feature is the emphasis on faithfully simulating a standard PROLOG environment. Communication with Cogent PROLOG is through a dialogue that consists of control predicates, goals, and terms. The top-level prompt (or main listener) expects the user to enter a goal. Cogent PROLOG attempts the proof of this goal and displays its results in the form of a dialogue on a continuously scrolling screen. A term containing an implication (*if* or :-) is assumed to be a rule and is automatically asserted into the data base.

Cogent PROLOG has several modes for adding clauses to the data base. The standard input file, *[user]*, creates a context for a series of terms to be added from the keyboard. Of course, the standard predicates *consult* and *reconsult* are available for adding clauses dynamically from a list. You can also consult a series of program files or libraries in a single statement.

The *comm* predicate allows access to DOS commands and permits an editor to be called from inside the PROLOG environment. Interactive debugging can be done by using the editor to modify a file, returning to the PROLOG environment, reconsulting the modified file, and testing the result with a new goal.

The Cogent PROLOG system kernel has all of the standard predicates. *Assert, assertz, clause, arg, retract, retractall,* and *op,* to name a few, are all made available by specifying that the standard PROLOG library be automatically consulted when the PROLOG environment is entered. In addition, you can create and specify private libraries that will also be automatically consulted at start-up.

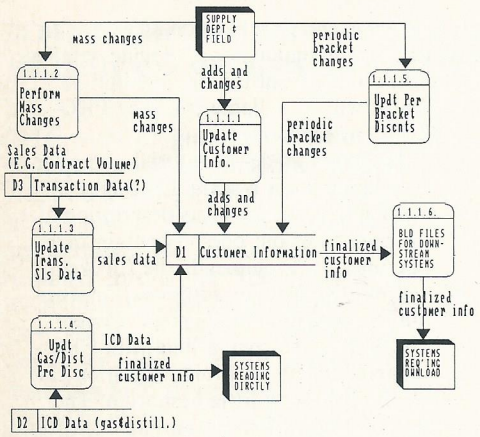
Cogent's implementation has no constraining data typing and also allows two or more predicates differing in arity to have the same name. These are all standard and essential features for the experienced PROLOG user.

Cogent PROLOG has the same debugging facility as is offered by traditional Clocksin and Mellish implementations. You can set spy points and trace the execution of each predicate as its proof is attempted. A leashing facility single-steps a program's execution. Cogent PROLOG has a "careful" mode that allows you to catch many errors of-

FREE CASE SEMINARS

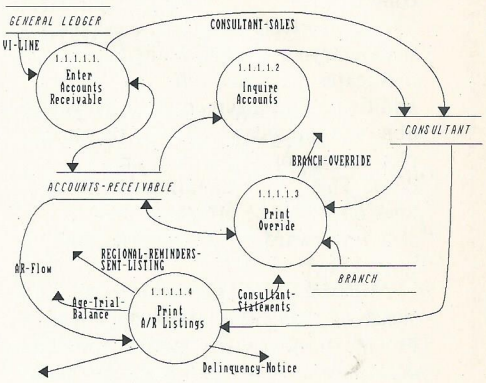
FEATURING THE VISIBLE ANALYST® WORKBENCH

TO REGISTER CALL (617) 969-4405



"THE VISIBLE ANALYST WORKBENCH IS A MOST IMPRESSIVE PRODUCT"
COMPUTER LANGUAGE MAGAZINE
January 1987

For the IBM PC,
XT, AT, 3270
and compatibles



Seminars will be held in most major cities in the U.S.

COME SEE WHY MAJOR COMPANIES ARE STANDARDIZING WITH US!!!

• Easy to Install • Friendly User Interface • Pricing • Easy to Learn •
• Interactive Data Dictionary • Expert Rules Analysis • Flexible Graphics •

STEP INTO CASE YOUR WAY FOR $595.00

Step 1 **DIAGRAMMING WITH A FUTURE: THE VISIBLE ANALYST®** **$595.00**
The right place to start for new system design or to document existing systems. Features include: Mouse Driven Graphics, Custom Symbols, Tree File Structured Projects, Constructs, Unlimited Nested Decompositions, Variety of Line Styles, Variety of Text Fonts, Block Move and More.

Step 2 **THE STRUCTURED UPGRADE: THE VISIBLE RULES** **$595.00**
Brings structured techniques to your company. Features include: Support for Yourdon or Gane & Sarson, Automatic Level Balancing, Consistency Checking, Error Listings and More.

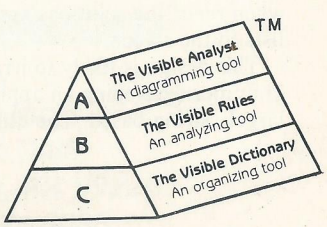
Step 3 **THE WORKBENCH UPGRADE: THE VISIBLE DICTIONARY** **$595.00**
Features: Fast Interactive Comprehensive Data Dictionary that populates automatically, Extensive Reporting Capabilities, ASCII Files and Formats provided for easy uploads.

Step 4 **THE LAN BASED WORKBENCH: COMING SOON**
Currently under development for the Novell Advanced Netware 286.

Step 5 **THE FUTURE WORKBENCH: UPDATES AND MAINTENANCE**
For just $50.00 per Workbench module per year you can insure that you stay tuned to the evolution in case technology.

THE COMMON SENSE CASE TOOLS ***FROM***
VISIBLE SYSTEMS CORPORATION
49 Lexington Street, Newton, MA 02165
SALES HOTLINE: (617) 969-4100 or 369-1800

THE LEADER IN LOW COST CASE TOOLS



CIRCLE 61 ON READER SERVICE CARD

ten associated with misspellings of predicate names.

A microcomputer language implementation wouldn't be complete if it didn't take advantage of machine-specific features. Cogent PROLOG has a number of built-in predicates that allow operations like writing to the screen, calling MS-DOS BIOS interrupts, reading the system clock, and executing DOS commands.

Cogent PROLOG's rich file-handling vocabulary supports the standard syntax predicates *see, seen, tell,* and *told.* In addition, modest screen handling is available with predicates to clear the screen, plot the cursor, and create simple windows. This screen-handling facility allows results to be presented in a straightforward but graphically interesting manner.

Although the Cogent interpreter is basically just that (an interpreter), it does provide an alternative method for storing data base files, which it calls fast load format. This is a proprietary file format that allows program files to be consulted at twice the speed of the standard ASCII format. However, the use of fast load files has absolutely no effect on the execution speed of applications.

Compatibility with the Clocksin and Mellish standard is critical to the usefulness of a language implementation. Cogent PROLOG is a faithful representation of the core syntax, with only a few major variations.

First, only one definition of an operator may be in force at any one time. This disallows the use of the minus sign as both a unary and binary operator in a single expression, for example.

Second, if a variable is instantiated to a term, then that variable may not be used as a goal in the body of a clause. The Cogent syntax requires that the predicate *Call(Term)* be used instead to evaluate the term.

Finally, the dot operator, a convenient functor for the manipulation of lists, is conspicuously absent.

Overall, Cogent PROLOG seems to be a well-designed package with a modest superset of Clocksin and Mellish PROLOG features. Compatibility with the standard syntax was a major consideration in Cogent's design.

The resulting product is somewhat lacking in the details that give a program aesthetic appeal. For example, strings are internally represented as lists of ASCII codes. Cogent PROLOG uses this format when reporting solutions and displaying lists to the screen.

This design is likely to irritate the programmer developing an application that's largely string-based. The difficulty can be overcome by avoiding string representations (expressions enclosed in double quotes) and developing applications that manipulate atomic representations (expressions enclosed in single quotes) instead. The library predicate *name* converts data from one representation to the other.

Functor names are essentially limited to eight characters since only those characters are considered when the interpreter parses an expression. This limitation could produce unexpected results if you're not careful with predicate names.

There's no access to a DOS window, although specific DOS commands can be executed at the top level or from within a program. The manual states that only external DOS commands (those with an extension of .COM and .EXE) can be executed. Actually, the procedure works with any DOS command-type file, including *DIR, COPY,* and *RENAME,* as well as batch files.

The current version of the interpreter does not handle floating-point numbers; arithmetic is limited to 16-bit signed integers.

The only facet of Cogent PROLOG's design that I question is the absence of automatic garbage collection. PROLOG is extremely memory-intensive. Whenever a procedure call is made, PROLOG must save information about the current environment, backtracking points, and variable bindings. Memory to save this information is obtained from several pools of storage known as the heap, local stack, control stack, and trail. When a clause is retracted, some memory can be reclaimed because this implies that the clause will no longer be needed in a proof.

The Cogent interpreter does not automatically reclaim memory that has been dynamically freed. Instead, it marks clauses that are no longer needed and hence are potentially reclaimable by stacks or the heap. Heap management is augmented by two predicates: *stats,* which returns the number of reclaimable heap cells, and *gc,* which performs actual garbage collection at a user's request.

This garbage collection can be performed during the execution of a program or at the top-level prompt. A large application that dynamically adds and retracts clauses from the data base with great frequency can experience memory allocation problems.

To circumvent this problem, a large program must contain additional code to properly monitor and manage the heap. Similarly, stack-intensive programs may run into difficulty, especially when tail-recursive procedures are used to excess. No checking is performed to eliminate tail recursion.

Although stack and trail sizes can be preset to their maximums, memory must still be available for the program, library, and text editor. For serious applications, a fully populated memory board is essential; the suggested minimum of 256K RAM will allow only the most rudimentary of programs. Despite these limitations, stack and heap overflows are handled in a civilized manner by informing the user of the need for garbage collection.

The Cogent package doesn't have an editor. The manual does provide detailed instructions on installing an editor so that it will be callable from the PROLOG environment.

The users guide—a tutorial and reference manual—is a scant 109 pages. The manual is adequate in its description of the predicates but doesn't contain many examples; it recommends the Clocksin and Mellish text for additional information on program development. Cogent PROLOG is so compatibile with Clocksin and Mellish that this suggestion can be accepted with little hesitation. The manual's lack of programming examples is ameliorated by the release disk, which contains approximately 50 sample programs. Technical support is available by telephone or mail directly from Cogent.

The Cogent interpreter is very much a product for the Clocksin and Mellish purist. For the experienced PROLOG programmer who regularly uses a standard standard implementation and isn't necessarily interested in flashy graphics or windows, Cogent PROLOG will be a satisfying experience.

The interpreter is an excellent educational tool in a setting where mainframe or minicomputer access is limited but the desire to teach or use the Clocksin and Mellish syntax is keen. Despite its limitations, Cogent PROLOG amply demonstrates that it is possible to adhere closely to a standard—and even go a little beyond.—J.V.

Logicware MPROLOG

MPROLOG's manual explains that MPROLOG is an abbreviation for Modular PROgramming in LOGic. Logicware's designers take modularity very seriously; the user must do likewise.

MPROLOG is designed to be operating-system independent, insofar as such a thing is possible. The user interface is to look the same whether you're using a PC or a CDC Cyber. Logicware states that MPROLOG runs on the Atari 1040ST, CDC Cyber, DEC VAXes, HP 3000, IBM mainframes, ISI, M68000 workstations (Apollo, Sun, and such), and Pyramid systems. While this does not mean you can't use features peculiar to a particular machine's operating system, it does mean that the integration of such features into an application is not as

Write for COMPUTER LANGUAGE!

November:
Special issue on algorithms
Deadline is August 1

December:
Programming in the 1990s
Deadline is September 1

Send your drafts or proposals on these topics or others of interest to programmers to:

Regina Starr Ridley
Editor
COMPUTER LANGUAGE
500 Howard St.
San Francisco, CA 94105

Now with dBx EXPRESS

C to dBASE

the dBx™ Translator

- C from dBASE II, III, III+ programs
- Move to UNIX, XENIX, QNX, MAC, AMIGA
- Faster, more reliable IBM PC programs
- Supports multi-user and network
- Run your code on any standard C system
- Know dBASE? Learn C easily.
- Priced from $350; available from distributors
- Includes full screen handler library
- Supports a choice of C database managers

from Desktop Ai

1720 Post Road East, Westport, CT 06880
Telephone: 203-255-3400 • Telex: 6502972226MCI
MCIMAIL-DESKTOPAI

dBASE is a trademark of Ashton-Tate dBx is a trademark of Desktop AI

CIRCLE 77 ON READER SERVICE CARD

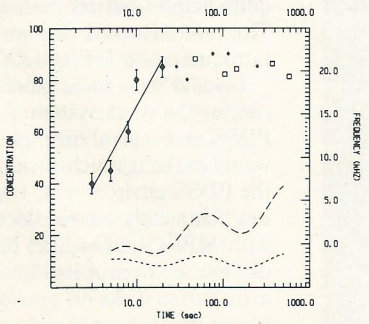
SCIENTIFIC/ENGINEERING GRAPHIC TOOLS
for the IBM PC and compatibles

FORTRAN/Pascal tools: **GRAFMATIC** (screen graphics) and **PLOTMATIC** (pen plotter driver)

These packages provide 2D and 3D plotting capabilities for programmers writing in a variety of FORTRAN/Pascal environments. We support MS, R-M, LAHEY FORTRAN and more. PLOTMATIC supports HP or Houston Instrument plotters. Font module available too!

Don't want to program? Just ask for **OMNIPLOT!** Menu-driven, fully documented integrated scientific graphics. Write or call for complete information and ordering instructions.

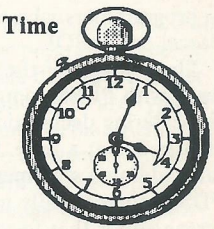
GRAFMATIC–PLOTMATIC–OMNIPLOT [S] & [P]



Microcompatibles, 301 Prelude Drive, Silver Spring, MD 20901
(301) 593-0683

CIRCLE 78 ON READER SERVICE CARD

Along With Your Computer, Your Time is the Most Important Thing You Own. . .So Why Waste It?



Quilt Programmer Productivity Tools will help you manage your software projects and get control of your time!

SRMS™
Software Revision Management System

- Full Featured Revision Control System
- All Versions stored in a Single ASCII File
- Support for Unlimited Libraries
- Support for all programming languages
- Allows you to use your current compilers and editors without conflict
- Windowing Shell interface to simplify use
- MERGE facility to consolidate different development paths easily, while pointing out conflicting areas
- Full audit trail tracking and reporting on all library components
- Handles big programming projects easily
- Full DOS Pathname and Environment Variable Support
- Requires DOS 2.1+, 224 K, F/H Disk

SRMS Version 3.0.....$185

QMAKE™
Intelligent Program Generation Utility

- Controls the rebuilding of even the most complex systems
- Relieves the developer of remembering which modules need to be rebuilt based on recent changes, how to rebuild them, and in what order to rebuild them
- Works with most compilers, assemblers, and linkers
- Supports full macro definitions, UNIX make-file compatability, recursive invocations, and command line parameters
- Interfaces completely with SRMS, providing you with a complete set of productivity tools to handle any size project
- Requires DOS 2.1+, 128K F/H Disk

QMAKE Version 1.2.....$99

SRMS + QMAKE $250

NEW ! TXT Tools NEW !

QSE - Quilt Text Stream Editor
QSRCH - Quilt File Search Utility (Like UNIX GREP)
QDIFF - Quilt Windowing File Difference Utility

TXTTOOLS Version 1.0..... $85

QUILT COMPUTING
7048 Stratford Road
Woodbury, MN 55125
(612) 739-4650

Volume Discounts and Dealer Inquiries Welcome

CIRCLE 85 ON READER SERVICE CARD

seamless as it might be in an implementation for a single machine.

The heart of the MPROLOG software development package is Logic-Lab, or program design support system (PDSS). As indicated, Logic-Lab is intended to support modular programming. An MPROLOG module is a collection of related predicates, some (or all) of which are visible from outside the module.

An MPROLOG program is made up of one or more modules, of which the main module has a goal. Modules can be activated or deactivated during the running of a program, allowing the user to create very large programs if desired. The separate modules can be linked together into a single program that can run separately using the interpreter.

The creation of a large MPROLOG program proceeds by first designing the program's module structure. (Top-down design is taken quite seriously.) This is best done on paper since it's essential to have a chart of a module's structure for working on it with PDSS.

A typical (small) module is shown in Listing 2. I took this module from the distribution disks; it's part of the suite of modules used for checking correct installation. (Incidentally, the installation section of the manual is exceptionally clear and easy to follow.)

The module's division into a declaration section and a body is based on Logicware's professed desire to make MPROLOG more friendly, particularly to programmers whose background is in Pascal or COBOL.

The contents of the module are entered with PDSS commands, which permit entering declarations or single clauses one at a time. If you make a mistake adding a statement to a module, PDSS automatically invokes a built-in screen editor that focuses on the single item being entered, an extremely useful feature during program construction. To change the predicate *test* in Listing 2, you can just type *edit test* at the PDSS prompt.

You can type in goals at the command-level PDSS prompt to test the module as you work on it. If an error occurs during an attempt to fulfill a goal, the trace mechanism is automatically invoked—another useful feature. Once you get the hang of working in PDSS (which requires a moderate investment of time), it's a very good environment. Since most of the the programs I wanted to run required minor adjustments to be compatible with MPROLOG, I was grateful for the system's capabilities.

Once you've debugged a module, a pretranslator creates a binary version of the module. These binary versions can be linked together using a consolidator program. The resulting linked binaries can then be run from the interpreter, bypassing the PDSS environment. Both the consolidator and interpreter require the creation of a special file to tell the program which files will be needed for the execution of a program.

This mechanism is handy since it serves as a primitive make utility. Unfortunately, the pretranslator, consolidator, and interpreter write their more useful error messages into a listing file. The error messages that do come to the screen are not particularly helpful.

This arrangement is probably a legacy from mainframe programming. In these days of interactive computing, you should be able to send errors both to the screen and a listing file. Listing a file for just one error is inconvenient. There didn't seem to be a significant speed difference between programs run from PDSS and those that had been translated into binary.

A large number of environmental parameters can be set for both the PDSS environment and stand-alone programs. Perhaps the most useful of these is the parameter *evaluation_limit,* which determines the number of statements that will be evaluated in trying to instantiate a goal. This parameter is very useful in protecting against runaway programs and is certainly more effective than trying to interrupt a program with Control-C.

MPROLOG is somewhat compatible with Clocksin and Mellish, though not completely so. (A version of PDSS, PDSS10, supports the standard predicates but doesn't support building stand-alone programs.) The principal differences between MPROLOG and straight Clocksin and Mellish implementations are that *assert* and *retract* are replaced by *add_statement* and *del_statement,* respectively, and *functor* and *args* are replaced by a single Swiss-army-knife predicate structure. Also, *call* is replaced by *evaluate.* Considering that portability between operating systems is one of the goals of MPROLOG, I found these changes gratuitous.

I was less disturbed by the changes made for the purpose of file handling. MPROLOG uses the concept of a channel for all I/O. Unusually, some of the I/O input predicates have versions that are undone on backtracking. MPROLOG does support direct-access files.

On the PC, MPROLOG only comes in an interpreted version. (Apparently, there is a compiler for larger machines.) Response time is quite adequate in the development environment.

I ran most of the benchmarks without significant modification. I did have to reduce the number of asserts from 1,000 to 500 to avoid running out of memory. Also, the recursive list walker produced a heap overflow that left me without enough room to edit.

While I was running the benchmarks, I discovered that floating-point arithmetic is not supported on the PC. The documentation gives the impression that it is, but a call to Logicware confirmed it isn't. However, you can call an external Pascal or assembler program if you need floating-point arithmetic.

In addition to the manual and disks that come with the professional version, I received a copy of a tutorial package that covers the fundamentals of PROLOG, rather than the use of Logic-Lab. I experimented with it briefly and found it quite helpful but somewhat inflexible. The tutorial would provide a reasonable introduction to MPROLOG.

I would have liked more tutorial material for the main system. A tutorial on PDSS and several example programs would make it much easier to understand the PDSS environment. Logicware promises completely revised documentation with MPROLOG v. 2.2; perhaps a tutorial will be part of it—I hope so. The distribution disks do provide some ready-to-run modules, including sample graphics and cursor manipulation routines and

```
module test_b.

/* Check if Argentina and Australia are the two countries */
/* in the southern hemisphere whose capitals are at the   */
/* same latitude. */

export( geographical_question/Ø ).
import( capital/2 , location/3 ).
visible( argentina , australia ).

/*$eject*/ body.

geographical_question :-test(X,Y), X=argentina, Y=australia .
test(COUNTRY1,COUNTRY2) :- capital(COUNTRY1,CITY1),
location(CITY1,LAT,_), LAT<Ø, capital(COUNTRY2,CITY2),
location(CITY2,LAT,_), CITY1@<CITY2 .

endmod /* test_b */ .
```

Listing 2.

Professional Programming Products

by BCSoft

FREE

PC-WRITE™ text editor
with every purchase.

PROGRAMMERS AND SOFTWARE DEVELOPERS - LOOK AT THESE PRODUCTS!

NEW

Quick-Tools™
for QuickBASIC™ users!

- A comprehensive library of over 90 subroutines and functions directly CALLable from your QuickBASIC programs.
- SORT routines allow sorting of single and multidimensional arrays.
- Binary search functions.
- Unique screen handling functions allow splitting screens, fast scrolling, special string printing, character attribute control, phantom cursors, etc.
- Keyboard scanning and status calls, with field inputing and decoding. On screen editing of input fields.
- Unique file handling routines.
- FREE Updates and phone support!
- Plus much, much more!

Only $129.95 Complete

NEW

NET-TOOLS™
NETBIOS Programming Tools

- NET-TOOLS allows you to write programs for ANY NETBIOS compatible local area network - fast and easily. CALLable from Microsoft Assembler, C, PASCAL, or FORTRAN.
- Add and Delete local names.
- Initiate and Cancel sessions. Just give NET-TOOLS the name of the computer you would like to call, and it will automatically locate the user and make the connection for you.
- Transfer Messages with automatic retries and error detection. Both the datagram and session protocols.
- Redirect Local Devices simply and easily with a single function call.
- Plus much MORE !
- Complete SOURCE CODE is provided, FREE !
- FREE Updates and phone support!

ONLY $149.00 Complete

TURBO.ASM™
For Turbo PASCAL users !

- A unique programming tool which is a must for every Turbo Pascal user. The only package designed for interfacing assembly language with Turbo Pascal.
- Outlines several different ways to add assembly language routines to Turbo programs, some without effecting your code space.
- Fully explains the internal workingsof Turbo Pascal and data passing methods.
- Includes a library of Assembly routines which can be used directly from Turbo Pascal.
- The best way to learn assembly language.
- FREE Updates and phone support!

Only $99.95 Complete

ASMLIB™
The Programmer's Library

- A set of over 210 subroutines to greatly increaseyour productivity. Written in assembly language anddirectly CALLable from Microsoft Assembler, C,PASCAL, and FORTRAN.
- Complete SOURCE CODE provided -FREE!
- Text WINDOWing functions allow up to 64 overlapping windows.
- TERMINATE and STAY RESIDENT programs arewritten easily. Programs can "POP UP" with a simplekeystroke. Use DOS calls from them, also.
- GRAPHICS on the EGA, CGA, and Hercules™ Monochrome.
- Virtual file functions.
- Full FLOATING POINT math and trig with 8087 support.
- Int. driven async. support, plus MUCH MORE!

Only $149.00 Complete.

NO ROYALTIES REQUIRED

To ORDER call or write to:

BC Associates
A division of BCSoft Corporation
3261 N. Harbor Blvd.
Suite B
Fullerton, CA. 92635

VISA, M/C, or COD orders are welcome!

CALL TOLL FREE 1-800-262-8010 in CA. dial 1-714-526-5151

CIRCLE 63 ON READER SERVICE CARD

list utilities.

The MPROLOG programming environment has many strengths to recommend it. The support for large programs is extremely good. The extensive support of modularity and information hiding do require a fair amount of time to get comfortable with; Logicware offers instructional seminars.

Everything you'll need to write high-quality PC applications is available, including graphics and support for exception handlers. Programs written in other languages (just Pascal and assembler currently) can be accessed. If command.com and your favorite editor are in the current directory, you can access DOS commands and the editor from within PDSS.

MPROLOG's weaknesses can all be traced to the laudable desire to make the package operating-system independent. As a result, the overall feel of the package is similar to mainframe programming rather than the more user-friendly atmosphere of personal workstation computing.

I found most of the error messages terse and uncommunicative. A typical response in the PDSS environment would be "XYZ wrong command —>." In spite of this, once I had adapted to the style of the package, it was quite pleasant to work with.

MPROLOG should be considered for large applications. It's reasonably close to the Clocksin and Mellish standard, and its limitations are few.—J.H.

Now COBOL Programmers Can Do Formatted Screens Quickly and Easily - with SCREENIO.

SCREENIO is the Screen Manager for Realia™ COBOL.

SCREENIO is a high-performance, COBOL-specific screen manager for the Personal Computer and compatibles. It's powerful, offers an impressive array of features, yet is easy to use. SCREENIO was written by experienced COBOL professionals and represents a truly COBOL approach to screen management on the PC.

Design and Revise Screens Painlessly With Our Panel Editor.

Use our interactive Panel Editor Facility to design your screens with a What-You-See-Is-What-You-Get approach. You can easily modify and experiment with screen layouts. Type titles and descriptive information on the screen, draw lines and boxes using the cursor movement keys, and paint the colors you want using the cursor keys or block functions. Use all 256 color combinations on your screens.

Define your data fields in COBOL terms; use your own data name, PICTURE and VALUE clauses. Specify the field type (various alpha and numeric types, hex, etc.), valid entries for the field (if you want SCREENIO to edit your data for you), and various other items. The Panel Editor will produce a copybook that contains this information. Panel revisions rarely mean changing your source code.

Programming for SCREENIO is Easy.

You can display your screen and accept data with as few as two statements; a COPY *panelname* in your WORKING-STORAGE SECTION, and a CALL SCREENIO statement in the PROCEDURE DIVISION. SCREENIO is linked with your application just as any other COBOL subroutine would be.

SCREENIO takes care of tabbing, data validation, and other functions expected of a good screen manager. When we return to your COBOL program, the data from the user fields on the screen is in your variables.

Yes, we've always done windows. SCREENIO supports Edit Masks, Automatic Error Detection and Editing, High Speed Video, Hot-Fields, Key Redefinition, Foreign Language Capability, Custom Tabbing Order, Color Control at Runtime, PC Speaker Control, and much, much more. A BMS Mapset Import is available. Most programmers are amazed at how easy it is to build flashy applications using SCREENIO and COBOL.

Our Support is Outstanding.

We provide superb telephone support. Upgrades are distributed to licensed users automatically for the first year. There are no Runtime Fees - the code you develop is yours to distribute freely.

Call us to discuss your application. Get a feel for the kind of quality support we provide.

We'll Send You a Free Demonstration Diskette.

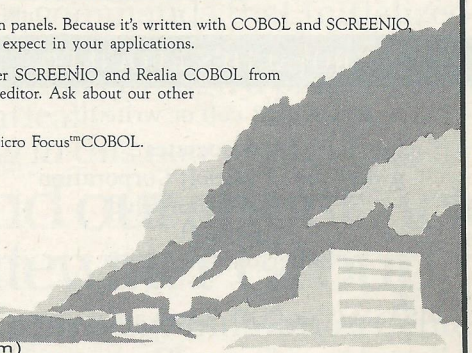
This limited version of our Panel Editor shows how you design panels. Because it's written with COBOL and SCREENIO, it also shows the kind of features and performance you can expect in your applications.

SCREENIO 2.1 is only $400 plus shipping. Save $100 - order SCREENIO and Realia COBOL from us. We also offer packages with Kedit, a great programming editor. Ask about our other programming tools, too.

Not satisfied with FORMS™? Ask about SCREENIO for Micro Focus™ COBOL.

In a hurry? Most orders are *delivered* within 24 hours!

- Amex MC Visa Check -



NORCOM

Northern Computing Consultants
P.O. Box 020897 • Juneau, Alaska 99802
(907) 780-6464 • TELEX 510 601 4951 (Norcom)

CIRCLE 64 ON READER SERVICE CARD

Expert Systems International PROLOG-2 Professional

Expert Systems International's PROLOG-2 Professional has extraordinary power. It's intended for the serious PROLOG programmer and is ideally suited for large-scale expert systems development.

The interpreter/compiler supports both the standard and Edinburgh syntaxes. PROLOG-2 provides a rich superset of the standard syntax features and complete access to the host-machine operating system.

The top-level listener is a dialogue consisting of queries, control predicates, and PROLOG programming statements. Debugging, editing, and error messages are provided in separate screen windows. Extensions of the DOS edit keys are available to the user; for example, F3 will retrieve the last command. Actually, PROLOG-2 maintains a small console stack making the last dozen or so commands available just by repeatedly pressing F3.

PROLOG-2's features are provided in discrete modules that can be included into the system at start-up time. Modules to support interactive editing, debugging, DEC-10 compatibility, a math coprocessor, grammar rules (definite clause grammar syntax), token processing, and on-line help are included by entering control statements into an initialization file. The initialization file is automatically consulted when the PROLOG-2 environment is started.

The PROLOG-2 system files are large and numerous; a hard disk is essential for efficient operation. You can successfully run PROLOG-2 from two floppy drives, although it's impossible for the floppies to hold both the help system and compiler at the same time.

The manual recommends that floppy drive owners make two copies of their system disks, one with the help system and the other the compiler system. The two systems can't be used concurrently during any one execution of PROLOG-2. It is possible to achieve reasonable (though not exceptional) performance from PROLOG-2 by using RAM disks. However, this advantage is only gained if the virtual memory is provided from any extended memory above the 640K recognized by DOS.

PROLOG-2 has an extensive vocabulary of built-in predicates that are compatible with the standard syntax. The full spectrum of arithmetic and comparison operators are available; transcendental functions are available only if the host machine is equipped with a math coprocessor. Multiple predicate declarations (functors with the same name but differing in arity) are allowed.

PROLOG-2 is one of the few PROLOG products that supports dot notation for lists. The dot is predefined as a prefix

operator with the following meaning: *.(a,b) <==> [a|b].*

The *op* predicate allows the dot to be defined as an infix operator. The dot operator is often seen in logic programming texts and is a feature of standard syntax that's rarely included in PC implementations.

Tail-recursive structures are predicates that call themselves; the recursive call is the last clause to succeed before a predicate is satisfied. A tail-recursive structure can be transformed into an iterative structure that doesn't require recursion.

By default, PROLOG-2 automatically optimizes to eliminate tail recursion, altering a program's backtracking. Tail-recursion optimization can be turned on or off by setting one of the PROLOG-2 state switches. Tail-recursion optimization is always on during compilation.

PROLOG-2 uses the standard *tell, telling, told, see, seeing,* and *seen* syntax in its file handling. It also supports creation of user-defined windows and I/O streams. PROLOG-2 manipulates random-access files and formatted input and output; bit-mapped graphics are not supported.

Operating system access is provided at two levels. PROLOG-2 has built-in predicates for many MS/PC-DOS commands. Commands are available for changing the logged drive or directory paths; creating or removing a directory; determining file sizes; and erasing, re naming, and determining the existence of files. The predicate *command/2* passes commands to the DOS command processor.

The operating system is also available from a separate host session window. The auxiliary session is invoked by a call of the form *command(_,N),* where *N* is bound to an integer. When a DOS session is finished, you can return to PROLOG-2 by typing *exit* at the DOS command prompt. The integer required in the *command* predicate specifies how much memory is to be freed by PROLOG-2 when it invokes the command.

PROLOG-2 claims all available system memory when it's running. Before it can invoke the auxiliary command processor, it must free up some of this memory by writing parts of itself out to disk. Since saving to disk is time-consuming, the user is given the option of choosing how much memory to free.

Unfortunately, when the *command* predicate is used in conjunction with session logging, an insidious bug crops up. Session logging allows a PROLOG-2 dialogue to be echoed into a file. The bug occurs when you start session logging, call an auxiliary DOS session, change the default drive, and then exit back to PROLOG-2.

Upon your return, PROLOG-2 fails to recognize the switch in drives. It continues to log the session, but it does so with the open file handle of the original drive. The results of this can be disastrous.

The first time this bug appeared, it wrote the session file onto the file allocation table (FAT) area of an unsuspecting floppy disk, completely and permanently trashing its data. The manual carries no warning about changing logged drives from an auxiliary session.

The effects of the bug can be avoided with appropriate care. By trial and error, I discovered I could assert an auxiliary clause to call DOS with the following definition:

```
new_command(YourSize) :-
        default_drive(X),
        command(_,YourSize),
        login(X).
```

This code saves the current drive, calls DOS, and then restores the current drive when it's finished. Apart from this one avoidable bug, PROLOG-2 appears to perform exceptionally well.

For PROLOG program development, debugging is extremely important. To use the trace facilities, the debugger system files must be loaded from the initialization file. The debugging system offers a standard tracing facility and lint checker. You can debug by leashing ports, setting spy points, or using single-step tracing. When an error is encountered in a query, the editor is automatically invoked, and you have the option of correcting, tracing, or aborting the goal.

PROLOG-2 supports grammar rules by inclusion of a directive in the initiali-

WITH REALIA, BE PREPARED FOR A FEW CHANGES IN YOUR PROGRAMMING STAFF

Give your applications developers Realia COBOL. Then stand back.

Realia COBOL brings the power of the mainframe right to your PC. You'll start saving time, money, and resources, whether you're maintaining an old system or creating a new one.

Realia COBOL offers the quickest compilation and the biggest file capacity of any PC compiler. 10,000 lines compile in 76 seconds. 10,000 records sort in 43 seconds. Best of all, Realia-compiled programs execute faster than any other PC compiler's. With RealCICS®, you can even handle online CICS programs.

At Realia, we also offer you something that has become a bit of a contradiction in terms: genuine support for a micro software product.

Realia COBOL—when time is of the essence.

REALIA®

10 South Riverside Plaza, Chicago, IL 60606 • (312) 346-0642 • Telex 332979

CIRCLE 65 ON READER SERVICE CARD

The Quality of the MKS Toolkit Speaks for Itself.

(Need We Say More?*)

"The **MKS Toolkit** is a terrific product! I depend on it every day."

"This is a very impressive piece of work and it can truthfully be said that it has made my PC-clone (NEC V-20 @8MHz) into a computer! There is little doubt that MKS is offering the package with the *most user power for the dollar* on the market today."

"I wouldn't be without the MKS stuff on a PC. And the documentation is superb! I use the MKS man pages to learn how to use **awk** and **sed** more efficiently on our BSD UNIX system at work!"

"The **Korn shell** is great!"

"This is a *solid* product. The program that makes my day is **cpio**, which allows me to move files to/from UNIX with minimum hassle and preservation of mod dates for *make* use."

"This program is a godsend for anyone who has to use both UNIX and MS-DOS . . . "

"I like the **MKS Toolkit** a lot. The idea of having the same editor on the PC and UNIX machines sure makes my life a lot easier."

" . . . at $139.00 the Toolkit is a bargain. . . .I hope you all reap the rewards of your virtue."

"The **MKS Toolkit** has provided me with UNIX capabilities I thought I lost when I moved to my PC."

"I'm impressed with the MKS tools, in particular with the breadth of what you provide. . . . I can see an order of quality and completeness in the MKS tools not found in the *PC/VI* package."

*These are unsolicited comments from MKS Toolkit users.

Price: $139

Now available in a separate package:

MKS VI

Price: $75

Mortice Kern Systems Inc.

43 Bridgeport Road East, Waterloo, Ontario, Canada N2J 2J4 **(519) 884-2251**

For information or ordering call collect.

Prices quoted in U.S. funds. MasterCard, VISA, American Express, and purchase orders accepted. OEM & dealer inquiries invited. UNIX is a trademark of Bell Labs. MS-DOS is a trademark of Microsoft Corp. Site-licensed to major American corporations. No UNIX licence required.

CIRCLE 72 ON READER SERVICE CARD

zation file. Grammar rule notation is a convenient representation for parsing sentences during natural language processing. When this feature is loaded, PROLOG-2 will automatically translate grammar rules into PROLOG clauses.

A similar feature is supplied for parsing tokens. Consider an input stream of a string of characters and numbers representing an arithmetic expression such as 3 + 5. Here, 3, +, and 5 are all tokens.

Using its token compiler, PROLOG-2 will create PROLOG clauses to recognize and parse constituent tokens from the stream. This facility is useful for developing a top-level PROLOG interpreter. The token and grammar facilities also provide convenient mechanisms for creating an expert system user interface.

PROLOG-2 supports modular programming, allowing efficient development of large-scale applications. The scope of predicates in individual modules can be controlled by declaring a functor either public or private. The private predicates' names are known only to the current module. Public predicates can be called by any module. The scope of a predicate can be changed dynamically (during execution of a program) with the built-in predicates *private(predname)* and *public(predname)*.

A single program or collection of modules can be compiled into what the manual calls assembly code, although the interpreter is still required for execution. Compilation is extremely slow, but the manual claims compiled files will execute five times faster than interpreted files. Once compiled, a program can't be altered or traced with the debugger.

External modules can be linked with compiled Lattice C and assembler code. C programs require an assembly language interface before they can be used by PROLOG-2.

For a modular programming project, a data module can be declared to reside in actual or virtual memory. This allows a program to exceed the size of actual system memory. Data modules are not well-defined in the users manual but presumably are collections of tailless assertions. When a module is in virtual memory, it will be swapped to and from disk when necessary. This facility increases the overall size of PROLOG-2 programs and makes them run more slowly.

The PROLOG-2 users manual is approximately 600 pages and packaged in two large ring binders. The manual contains an index, glossary, and 200-page catalogue of built-in predicates. The manual doesn't have a tutorial and assumes more than an elementary knowledge of PROLOG on another system.

Despite its girth, the manual is very poorly organized. Learning to use some of the features from the manual is like trying to learn a foreign language from a dictionary. It pays very little attention to how predicates work together; for its 600 pages, there are probably less than a dozen complete examples.

The treatment of built-in editor, windows, and streams suffers most from the lack of examples. Here's an example of the manual's presumptive tone from the "Opening Streams" section: "Opening a stream does not make it current. A stream must be made current before input or output can take place." How is a stream made current? There's no discussion of current streams; the manual's next topic is closing a stream.

Using the editor for the first time is equally frustrating. Again, there's no complete example of how to invoke it. The answers to these basic questions can only be gleaned through trial and error or technical support.

PROLOG-2 is distributed on five disks that hold the PROLOG-2 system, run-time modules, help system, compiler, and sample programs. In contrast to earlier releases, Expert Systems International has removed all copy protection.

Only a few sample programs are supplied. However, one of the programs is an impressive LISP interpreter written entirely in PROLOG. Examples of external C programs callable from PROLOG-2 are also provided.

PROLOG-2 is well suited for large-scale expert system development. It isn't cheap, but its performance and features make it worth the price.—J.V.

Borland International Turbo PROLOG

In the letter that comes with Turbo PROLOG, Philippe Kahn, the president of Borland International, explains that Turbo PROLOG is "the first PROLOG implementation for IBM PCs and compatibles to deliver a high level of performance while remaining conservative in its memory requirements." Apparently, Borland decided that the best way to do this was with a conventional compiler-type mechanism—a choice that has had a considerable influence in shaping the package's programming environment.

Following the approach taken in Turbo Pascal, Turbo PROLOG presents a complete environment for developing and running programs. The default screen provides four windows: an edit window for creating the program, a dialogue window for run-time communication with programs, a message window that tracks the progress of the current PROLOG session, and a debug window for tracing a program's execution. A user can reconfigure the windows permanently or just for the current session.

The system makes use of menus for such functions as loading files and examining directories. A typical session re-

JACK™ The Resident Program Developer's Kit.

JACK™ contains everything you need to create your own RAM resident software without the headaches of memory management, windows or DOS re-entrancy.

JACK™ FEATURES:

Virtually any C or ASSEMBLER program can be made memory resident without any modification to the existing source code. All you have to do is link your program with our library.

Screens are automatically saved to conventional memory [or the expanded memory of an Intel Above Board if present] when popping up and restored upon popping down .

DOS function calls and disk I/O within your memory resident program can safely be used and JACK™ does not make any use of undocumented DOS calls to solve the DOS re-entrancy problem.

All programs developed with JACK™ peacefully coexist amongst themselves and with the many popular pop-up utilities from other developers.

CrackerJack would like to introduce BEYOND™, a library of memory management and service routines for the Intel Above Board. Now all of your programs can take advantage of expanded memory. With our library and Intel's ABOVE Board, your programs can go ABOVE and BEYOND™ their normal limitations.

To Order, please send $199.95 [US] + S/H for JACK™ or $49.95 [US] + S/H for BEYOND™ by certified check or international money order to:

Crackerjack Microsoftware Corporation
200 Bay Street PO Box 86
Toronto, Ontario M5J-2J2
Canada

(416) 865-9621.

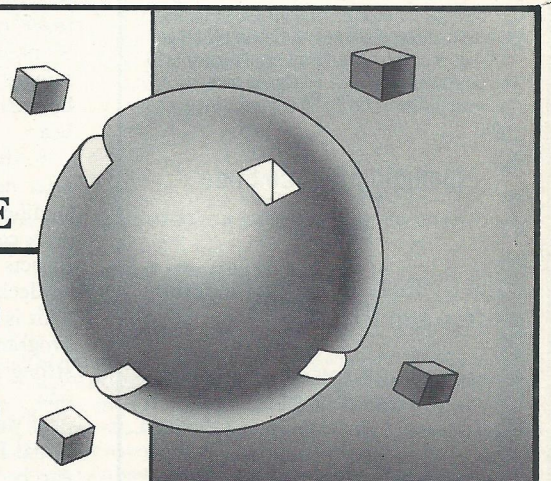
VISA

Please Note: When ordering, please specify either the Assembler version or C version of JACK™ and indicate the vendor and release level of your compiler.

JACK™, CRACKERJACK and BEYOND are registered trademarks of CrackerJack Microsoftware Corporation, Intel is a registered trademark of Intel Corporation, Microsoft and MS-DOS are registered trademarks of Microsoft Corporation.

CIRCLE 66 ON READER SERVICE CARD

C PROGRAMMERS! THE TOOLS YOU NEED AT A PRICE YOU'LL LIKE



BTree Supports all index file operations. Very quick sequential or random access, duplicate keys, multiple indices, fixed and variable length data records are all supported. **75.00**

ISAM Works on top of BTree to provide a simple, yet powerful application program/file system interface. Complex filesystem manipulation becomes a snap. Provides the power of a database manager with the flexibility of a programming language. **40.00**

lp Finally, a completely device independent printer library! lp drives any printer as accurately as possible and allows easy access to its most sophisticated features. Multiple fonts, multi-column output, complex margin formatting, and much more. Pays for itself the first time it's used. **75.00**

Snake The ultimate 'make' utility. We couldn't find a good one, so we wrote a great one. Has all kinds of powerful features including wild card filename expansion, nested macros, and multiple dependency and rules definitions. Ready to go for MS-DOS; C source is there if you use another operating system. **59.00**

Combine & Save: BTree + ISAM + lp **149.00** + Snake **199.00**

Each product includes a typeset manual, example programs, and complete C source code that runs on any operating system. Softfocus products may be incorporated into applications royalty-free.

Credit card orders accepted. Visa, M/C, Amex. Dealer inquiries invited.

softfocus
1343 Stanbury Drive
Oakville, Ontario, Canada
L6L 2J5
(416) 825-0903

CIRCLE 67 ON READER SERVICE CARD

CQL

SQL Compatible Query System adaptable to any operating environment.

CQL Query System. A subset of the Structured English Query Language (SEQUEL, or SQL) developed by IBM. Linked files, stored views, and nested queries result in a complete query capability. File system interaction isolated in an interface module. Extensive documentation guides user development of interfaces to other record oriented file handlers.

Portable Application Support System

Portable Windowing System. Hardware independent windowing system with borders, attributes, horizontal and vertical scrolling. User can construct interface file for any hardware. Interfaces provided for PC/XT/AT (screen memory interface and BIOS only interface), MS-DOS generic (using ANSI.SYS), Xenix (both with and without using the curses interface), and C-library (no attributes).

Screen I/O, Report, and Form Generation Systems. Field level interface between application programs, the Query System, and the file system. Complete input/output formatting and control, automatic scrolling on screens and automatic pagination on forms, process intervention points. Seven field types: 8-bit unsigned binary, 16 bit signed binary, 16 bit unsigned binary, 32 bit signed binary, monetary (based on 32 bit binary), string, and date.

Including Source Code

$395.00

File System interfaces include C-tree and BTRIEVE.

HARDWARE AND FILE SYSTEM INDEPENDENT

KURTZBERG COMPUTER SYSTEMS

41-19 BELL BLVD.
BAYSIDE, N.Y. 11361

VISA/Master Charge accepted
(718) 229-4540

*C-tree is a trademark of FairCom

IBM, SEQUEL, PC, XT, AT are trademarks of IBM Corp.
MS-DOS and Xenix are trademarks of Microsoft Corp.

CQL and the CQL Logo are trademarks of Kurtzberg Computer Systems.

CIRCLE 50 ON READER SERVICE CARD

quires loading a program file, followed by some editing; the program is then run from the top-level menu. Since Turbo PROLOG is compiled when a program is run, checks are made before any attempt to instantiate a goal. If the program fails any of these checks, control is passed to the editor with the cursor at the point where a check failed. This is a considerable convenience during program development; users of Turbo Pascal will be familiar with its advantages.

Although the manual claims that Turbo PROLOG conforms to the Clocksin and Mellish standard, there are significant differences between the syntax of the two—most of which can be attributed to the choice of a compiled language. Since these differences have been the subject of some debate, I'll spend some time discussing them so that you'll know what to expect from Turbo PROLOG. (For an earlier review of Turbo PROLOG, see "Inside Turbo PROLOG," AI Eye, *COMPUTER LANGUAGE*, July 1986, pp. 23-28.)

The most obvious difference is that Turbo PROLOG requires that all predicates be declared before they are used. I can illustrate this with Turbo's version of the member program (Listing 3). This example is taken directly from the program disk. I've added a goal to illustrate where it would appear in a complete program.

Domains are Turbo PROLOG's name for types; the standard ones are *char, integer, real, string, symbol,* and *file.* Symbols are handled differently from strings: they go into the symbol table, but strings are accommodated on the heap.

In the preceding code, the asterisk after *name* in the definition of *namelist* signifies that *namelist* is a list of names. As a consequence, predicates that handle objects from different domains need to be declared for each of those domains. This is an unfamiliar restriction in AI programming but, given the benefits of strong typing, perhaps not an unwelcome one.

If you're familiar with more conventional PROLOGs, you'll notice that *if* can be used instead of :-. *or* can also be used instead of a semicolon, and *and* instead of a comma. The manual's examples use both sorts but not within a single program. The use of *and, or,* and *if* does make programs more readable.

Turbo PROLOG's other major area of divergence from more conventional PROLOGs is its use of *assert* and *retract.* Only facts, not clauses, can be asserted and retracted. Predicates that are to be the subject of assertion and retraction must be declared in a special declaration section introduced with the reserved word *database.* Predicates that manipulate the data base must be typed so that they refer to the reserved domain *dbasedom.*

With this design, Borland has elected to view PROLOG as a language for data base access rather than a language for writing systems that learn new rules. Applications requiring intelligent data base access will benefit from the forthcoming Turbo PROLOG Toolbox, which is promised to contain predicates for file transfers from Lotus 1-2-3 and dBASE.

While this data base design does restrict the kinds of self-modification a program can do, it is in line with modern language theory. For similar reasons (related to the compilation method), you can't define operators and their precedence.

Most of Edinburgh PROLOG's other features are available in some way through the programming environment. Also, Turbo PROLOG provides numerous useful predicates not found in Clocksin and Mellish.

Turbo PROLOG has a very full set of predicates for file handling (although not the *see* and *tell* family). These predicates permit the creation and use of random access files by a PROLOG program. Predicates are provided for the manipulation of windows in both character and graphic modes—even turtle graphics.

You can invoke DOS commands and BIOS functions from a Turbo program and even link programs written in assembler, C, FORTRAN, or Pascal. (Ironically, since the current version of Turbo Pascal can't create .OBJ files, Turbo Pascal programs can't be linked to Turbo PROLOG programs. Borland promises to rectify this with Turbo Pascal v. 4.0.)

Some tools are available for modular programming. A project definition is a file with the names of modules to be linked together. Objects are shared between modules solely through the use of global domains and predicates, so there's only limited support for information hiding.

The Turbo PROLOG package comes with two disks and a manual. One disk has the PROLOG system, and the other has examples. Installation is straightforward. The manual's installation section

```
domains namelist = name* name = symbol predicates
member(name,namelist). goal member(jim,[fred,anne,jim]). clauses
member(Name,[Name|_]). member(Name,[_|Tail])
if member(Name,Tail).
```

Listing 3.

appears somewhat belatedly in chapter 12. If you're willing to accept the defaults, all you'll have to do is copy the system disk and type "PROLOG" at the MS-DOS prompt for the directory containing the system.

If you've never used PROLOG before, the manual has an excellent tutorial (with the understanding that this PROLOG is very different from conventional interpreted versions). All the programs in the text are also on disk, along with answers to the exercises. (A minor point: The C program and its driver in the text are not the programs on disk.) I ran several of the sample programs and found that they generally worked as advertised.

The manual's programmer's guide has a number of useful tips on programming style, several of them helpful in making programs more efficient. The compiler has some warnings built in that catch such things as variables that are only used once (you can ignore the warning if you want).

So how fast is Turbo PROLOG? The examples compiled in a matter of seconds. I didn't compile any large programs, so I can't report on that aspect.

As to speed of execution, Turbo PROLOG is fast. On the benchmark suite, performance on arithmetic predicates was particularly impressive.

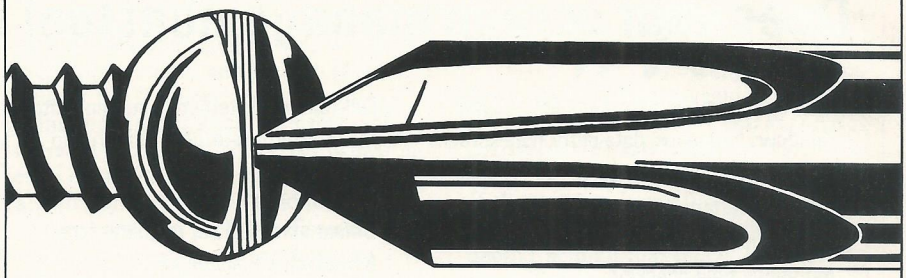
The modification of the benchmark programs to run under Turbo PROLOG took some time. Many of the benchmarks have a large number of predicates, and while Turbo PROLOG will accept up to 500 predicates, adding declarations for 100 predicates was tedious. (Perhaps I should have written a PROLOG program to write the declaration sections automatically.)

Turbo PROLOG is a well-crafted package. Initially, I was somewhat resistant to Borland's compiler approach, its restrictions on handling predicates from different domains, and the inability to assert and retract rules. For certain AI applications, these would be real limitations.

On the other hand, for data base applications, these compromises are not serious and may in fact lead to better programs. The ability to link together separate modules and even access programs in other languages is desirable for writing large programs, although these features are somewhat marred by the requirement that shared predicates be global.

Turbo PROLOG provides a wealth of predicates beyond the usual Clocksin and Mellish ones, making the package a very attractive programming environment. The compiler approach does succeed in blending some of compilation's advantages with PROLOG's declarative strengths. Applications programmers can write the window-rich user interfaces

ISN'T IT A PITY...



Everything Isn't As Accommodating As

c-tree™ / **r-tree**™

FILE HANDLER — REPORT GENERATOR

Performance and Portability

For all the time you devote to developing your new programs, doesn't it make sense to insure they perform like lightning and can be ported with ease?

c-tree: Multi-Key ISAM Functions For Single User, Network, & Multi Tasking Systems

Based on the most advanced B+ Tree routines available today, **c-tree** gives you unmatched keyed file accessing performance and complete C Source Code. Thousands of professional C programmers are already enjoying **c-tree**'s royalty-free benefits, outstanding performance, and unparalleled portability.

Only FairCom provides single and multi-user capabilities in one source code package, including locking routines for Unix, Xenix, and DOS 3.1., for one low price! In addition, **c-tree** supports fixed and variable record length data files; fixed and variable length key values with key compression; multiple indices in a single index file; and automatic sharing of file descriptors.

r-tree: Multi-File Report Generator

r-tree builds on the power of **c-tree** to provide sophisticated, multi-line reports. Information spanning multiple files may be used for display purposes or to direct record selection. You can develop new reports or change existing reports without programming or recompiling and can use any text editor to create or modify **r-tree** report scripts including the complete report layout. At your option, end users may even modify the report scripts you provide.

Unlimited Virtual Fields; Automatic File Traversal

r-tree report scripts can define any number of virtual fields based on complex computational expressions involving application defined data objects and other virtual fields. In addition, **r-tree** automatically computes values based on the MAX, MIN, SUM, FRQ, or AVG of values spread over multiple records. **r-tree** even lets you nest these computational functions, causing files from different logical levels to be automatically traversed.

Unlike other report generators, **r-tree** allows you to distribute executable code capable of producing new reports or changing existing reports without royalty payments, provided the code is tied to an application. Your complete source code also includes the report script interpreter and compiler.

How To Order

Put FairCom leadership in programmers utilities to work for you. Order **c-tree** today for $395 or **r-tree** for $295. (When ordered together, **r-tree** is only $255). For VISA, MasterCard and C.O.D. orders, call 314/445-6833. For **c-tree** benchmark comparisons, write FairCom, 2606 Johnson Drive, Columbia, MO 65203.

Complete C Source Code & No Royalties!

Xenix is a registered trademark of Microsoft Corp. Unix is a registered trademark of AT&T.

CIRCLE 51 ON READER SERVICE CARD

▪ WINDOWS ▪ MENUS ▪ DATA ENTRY ▪ SCREENS ▪

HI-SCREEN XL™

Does your screen management tool include all of this?

- ☐ Unlimited number of screens & windows
- ☐ Windows for menus, data entry, help screens
- ☐ Pop-up, pull-down and Lotus-like Menus
- ☐ Up to 26 overlapping windows on each screen
- ☐ On-line help management
- ☐ Automatic field checking (type/range/format/etc.)
- ☐ 2 modes for input: field by field/full screen
- ☐ Scrolling capability
- ☐ Full-featured interactive screen editor
- ☐ Move/copy/delete/color/draw/undo functions
- ☐ Merge feature
- ☐ Online extended set of characters
- ☐ Redefinable function keys
- ☐ Data entry test mode under the editor
- ☐ Direct access to DOS from the editor
- ☐ True ease of use

...and the ability to:

- ☐ Capture screens from other applications
- ☐ Call screens for reference from your text editor
- ☐ Modify screens without recompiling your program
- ☐ Trace/debug compiled programs
- ☐ Display screens from RAM memory
- ☐ Use screens & windows from DOS and batch files
- ☐ Print screens with clear data field description

...and does it do all this for all versions of most popular programming languages?

If it does, Congratulations: you must be a **HI-SCREEN *XL*™** user! If it doesn't, call or write NOW to order your copy of **HI-SCREEN *XL*™**!

Softway, Inc., PC/Soft Product Line 500 Sutter St. Suite 222 San Francisco, CA 94102. Tel. (415) 397-4666. **HI-SCREEN *XL*™** is $149.00. S&H $5.00

▪ for PASCAL, BASIC, C, COBOL, dBASE, FORTRAN, ETC...

CIRCLE 62 ON READER SERVICE CARD

Introducing...

The Software Developers Card Deck

Reach 100,000 software developers who subscribe to COMPUTER LANGUAGE, AI EXPERT, and UNIX Review

Target Your Audience
For the first time ever, you can send a card directly to software developers. Your card goes directly to our subscribers—all individuals who have paid or qualified for their subscription.

It's Affordable
Our great introductory offer makes this a special value for advertisers. Reach three growing marketplaces for one low price!

The Software Developers Postcard Deck
Mailing date: September 1987
Reservation Closing: August 17th

For advertising rates, postcard specifications, and to reserve space for your card, contact your sales representative today:

Boston	Ohio	New York	San Francisco
Linda Pizzotti	Gina Dalpra	Robyn Perry	Stephanie Dale
(617) 232-5470	(216) 493-8733	(212) 683-9294	Susan McDonald
			(415) 397-1881

that have become the norm for user-friendly programs.

One final note: the terms of the software license are written in admirably clear English. I'm not a lawyer, but I found the statement to be both generous and unambiguous. Borland is to be commended on this.—J.H.

Automata Design Associates VML PROLOG

VML PROLOG is an extensive, professional-level PROLOG system suitable for research and development. The system provides a rich programming environment with a large set of powerful and innovative extensions.

The package reviewed included a loose-leaf manual and four noncopy-protected disks. Versions of the interpreter are included for systems both with and without a math coprocessor. Example programs are included in compressed format.

Installation consists of copying the files to the destination directory and running a configuration program to set the amount of memory allotted to the interpreter workspace.

Though voluminous, VML PROLOG's documentation lacks the organization and structure you'd expect from a $200 product. Granted, it's difficult to present a coherent view of such an exhaustive series of functions and capabilities, but the inclusion of an index would be welcome.

VML supports a vast array of built-in predicates and operators, including bitwise operations, definite clause grammars, inference engines, and meta interpreters. This array of extensions is staggering, though at times oddly inconsistent in its coverage of a particular facility. For example, the set of bitwise operators does not include a shift operator, leaving you to perform arithmetic operations to achieve the desired effect. While this is a solution, it obscures the intent of a computation that would otherwise be lucid. This shortcoming is balanced by the innovative extensions VML makes to PC-based PROLOG.

One area of innovation is the introduction of global variables. While standard PROLOGs require that values to be maintained over successive goals be asserted into the data base, VML permits the definition of global variables that maintain their values. This saves time and reduces tedious bookkeeping.

Global variables are set using the infix *set* operator and can hold structures or numbers. Global variables can also be used to store arrays as actual arrays of memory locations rather than less space-efficient lists.

Very large values can be kept in special variables known as superatoms. Superatoms are functors whose names may be as long as 3,000 characters. Using superatoms, you can store entire screens, complete with cursor sequences, and display them by simply printing the value of the functor.

The standard PROLOG flow of control can be extended with an intriguing suite of VML predicates. *Cond,* a conditional form of *cut,* allows conditional removal of successful goals from the stack upon backtracking. Complementing *cond* is *repcond,* essentially a conditional *repeat* predicate. *if* and *repif* conditionally undo or repeat instantiations made by a successful goal.

The standard PROLOG data base model's flexibility is extended with domains. VML's domains are programmatic modules in which all external interactions must occur through a single predicate called the port. All other predicates are inaccessible to external domains.

Ports are either exports or imports, allowing predicates defined in the domain to be available to external callers or externally defined predicates to be available inside the current domain. This is analogous to Common LISP in the sense that it facilitates the modularization of PROLOG programs, allowing multiple programmers to work together with greater ease.

Domains are scoped in an inheritance structure, allowing lower-level (child) domains access to data and predicates in their parent domains.

VML has a well-thought-out virtual memory system that can overcome the PC's memory limitations. Virtual memory is available through the special *consultv* predicate. *consultv* caches portions of the data base in memory while retaining indexes to the rest of the data base, which is kept in an external file. This method allows data bases several times larger than available memory to be incorporated into a system with relative ease and only a small penalty in performance.

VML PROLOG includes an extensive set of IBM PC-compatible functions, including full keyboard, graphics, timer, and screen support. Timer support includes the implementation of an asynchronous timer interrupt, suggesting the intriguing possibility of background and foreground tasks and other sorts of multitasking. You can set a timer trap to transfer to a specified goal upon expiration of the timer. In cases when a desired DOS operation is not built-in, predicates are provided to generate the necessary interrupt instructions.

The package provides a complete tracing and debugging system, including a useful option for diverting trace information to a disk file for later review. A se-

Add Data Security to Your C Programs

The SECURITY LIBRARY™

Add Fast or Thorough Encryption or Compression to Your Programs WITHOUT Royalties

Build the safest and most popular methods of protecting your data directly into your program without paying royalties. Don't worry if you're not sure what to use. The discussion of security methods in the manual (with demos on disk) will help you make a choice based on your application.

With Security Library you can:

- Keep files secure on a multi-user network
- Speed up data transmission and communications through data compression
- Control access based on privileges given to a hard disk or LAN user

"I create custom software for business applications. I'm using the Security Library to encrypt so that certain information will not be readily available. The documentation is excellent. It gives a thorough explanation about how you can secure a file."

— Bruce Philips, Custom Software Design
Virginia Beach, CA

Requires MSDOS 2.0+. Works with Microsoft C and Computer Innovations' C86. Please specify compiler when you order.

You Choose the Security Level

Algorithms provided include The National Bureau of Standards' Data Encryption Standard (DES) and the Vernam and Vegenere ciphers. Encrypt a 10K file in 3 seconds with one method or 50 seconds with another.

Six algorithms are provided, along with password and non-password encryption schemes.

Valuable Extras

The Huffman coding routine can *reduce the size of a file* by 25 to 50%. The routine to change the attribute bytes of a file can make that file invulnerable to casual browsing or accidental deletion. There's even a program to *change every byte of a file to a null character* - not even un-erase programs can recover it then!

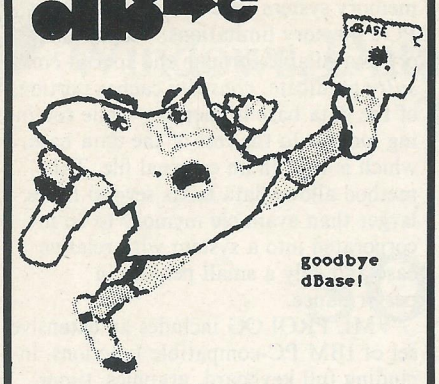
Call 800-821-2492 to order Security Library risk-free for only $125. Source & Object is $250.

Solution Systems™
335-L Washington St.,
Norwell, MA 02061
(617) 659-1571

Full refund if not satisfied in 30 days.

CIRCLE 58 ON READER SERVICE CARD

dB→c



dBASE Programmers

are you ready?

dB→c

If you need it, and think you can handle it, dB2c is here now.

dB2c Offers:

- Complete implementation of dBASE III+ functions and commands in C source code.
- Interfaces for Lattice's dBC and Faircom's c-tree.
- Complete source code.
- Support for the following C Compilers: Microsoft, Lattice, Aztec, DeSmet, and Instant-C.
- The freedom and peace of mind that only source code can offer.
- Use of familiar syntax while learning C.
- Works on color, monochrome, and EGA monitors.
- Versions available for MS-DOS, Xenix, Unix, OS-9 and Concurrent DOS.

dB→c Toolkit $299.

DAVID J. MARSH

Call or Write:

SOFTWARE CONNECTION, INC.
POB 712, Ely, MN 55731
(218) 365-5097

CIRCLE 70 ON READER SERVICE CARD

ries of control sequences provide access to a pop-up break facility any time during program execution. This facility includes on-the-fly redirection to trace files, toggling printer output, and single-stepping execution—each of which is performed using the common box debugging style.

Expert system applications using belief revision or evidential reasoning are supported with a unique theory facility. This extension allows entire sets of clauses to be grouped and manipulated as a single entity. The set can be altered, tested, and destroyed as a single, monolithic unit, allowing a number of hypotheses to be tested out and either asserted or denied under programmatic control. This feature has interesting implications for planning and scheduling applications.

One of VML's most unique features is the ability to manipulate cyclic data structures. This ability is accessible through an alternative unification mechanism that can be selectively enabled in the interpreter. The mechanism includes cyclic versions of the unifier algorithm, *cut* operation, and term-printing logic. The VML literature doesn't cover this feature fully, but applications using circular queues and self-referential lists would clearly benefit.

VML PROLOG is a powerful development environment that offers considerable flexibility. The library of built-in predicates is sizable and appears to be constantly increasing, though coverage of functionality is at times frustratingly incomplete. VML PROLOG has much innovative functionality, offering everything from ways to better utilize available storage to high-level support of reasoning capabilities.—S.E.

Steven Engle is an artificial intelligence software engineer for Expert-EASE in Belmont, Calif. His interests include robotics, computer vision, and neural computing.

Jonathan Hodgson is a professor of math and computer science at St. Joseph's University in Philadelphia, Pa., doing research in operating systems and artificial intelligence.

James Vita is a programmer at City College of New York, New York, where he designs mainframe data base systems. He is currently completing his M.A. in computer science and is interested in object-oriented languages and computer vision.

PROLOG products

ALS PROLOG
Professional version—$499
Personal version—$199
Applied Logic Systems
P.O. Box 90
University Station
Syracuse, N.Y. 13210
(315) 471-3900

Arity/PROLOG
Compiler and interpreter—$795
Interpreter—$350
Arity Standard PROLOG—$95
Arity Corp.
358 Baker Ave.
Concord, Mass. 01742
(617) 371-2422

Chalcedony PROLOG/i—$99.95
Chalcedony Software Inc.
5580 La Jolla Blvd., Ste. 126
La Jolla, Calif. 92037

Cogent PROLOG
Compiler and run-time system—$200
Interpreter package—$79
Cogent Software Ltd.
21 William J Hts.
Framingham, Mass. 01701
(617) 875-6553

MPROLOG
Full production version—$495
Version without pretranslator and consolidator—$195
Tutorial package—$49
Logicware Inc.
5915 Airport Rd., Ste. 2000
Mississauga, Ont.
Canada L4V 1T1
(416) 672-0300

PROLOG-2 Professional—$895
Expert Systems International
1700 Walnut St.
Philadelphia, Pa. 19103
(215) 735-8510

Turbo PROLOG—$99.95
Borland International
4585 Scotts Valley Dr.
Scotts Valley, Calif. 95066
(408) 438-8400

VML PROLOG—$200
VMA PROLOG—$250
VMI PROLOG—$99.95
FS PROLOG—$49.95
Educational version—$29.95
Public domain version available
Automata Design Associates
1570 Arran Way
Dresher, Pa. 19025
(215) 646-4894

CANADA'S SOURCE FOR C

– Canadian Sales
– Canadian Service
– Canadian Technical Support
– Canadian Product Knowledge

We specialize in programming & development software

LIFEBOAT • LATTICE • GREENLEAF • PHOENIX
SOFTCRAFT • MICROSOFT • BLAISE • ESSENTIAL
AGE OF REASON • DESMET • AZTEC
MARK WILLIAMS • GIMPEL • ROUNDHILL • GSS
HALO • FAIRCOM • RAIMA • INTEL • etc. • etc. •

VISA Call for full price list—Dealer enquiries welcome MasterCard

We know our products — we use them!

SCANTEL SYSTEMS LTD.

801 York Mills Rd., Don Mills, Ont., M3B 1X7
(416) 449-9252

CIRCLE 73 ON READER SERVICE CARD

SOFTWARE DEVELOPMENT '88

SAN FRANCISCO, FEBRUARY 17–19, 1988

The 1st annual conference
***exclusively* for software developers!**

The publishers of COMPUTER LANGUAGE, AI EXPERT and UNIX REVIEW have joined together to expand on the successful COMPUTER LANGUAGE C Seminar. We plan to create the most practical and useful programming seminar and trade show possible. Become involved in this event from the start by filling out this coupon today.

Name _____

Company _____

Address _____

City, State, Zip _____

Phone _____

☐ **Please put me on your mailing list** to receive further information on attending the seminar and trade show as it becomes available.

☐ **I may be interested in exhibiting at the trade show.** Please send me further information as it becomes available.

☐ **I would be very interested in these seminar topics:**

_____

_____

☐ **I suggest the following speakers:**

_____

_____

☐ **I would like to present a paper about:**

_____

Send to: Beatrice Blatteis, *Software Development '88,* Miller Freeman Publications, 500 Howard Street, San Francisco, CA 94105.

Make the Logical Choice

Springer-Verlag for AI

Visit Booth #203 at AAAI '87, Seattle

The Knowledge Frontier
Essays in the Representation of Knowledge
Edited by **Nick Cercone** and **Gordon McCalla**
1987/Approx 543 pages/93 illus/Cloth $40.00 (tent.)
Symbolic Computation
ISBN 0-387-96557-2

Natural Language Parsing Systems
Edited by **Leonard Bolc**
1987/461 pages/Cloth $59.50 (tent.)
Symbolic Computation
ISBN 0-387-17537-7

Computational Models of Learning
Edited by **Leonard Bolc**
1987/243 pages/Cloth $65.50 (tent.)
Symbolic Computation
ISBN 0-387-16318-2

On Knowledge Base Management Systems
Integrating Artificial Intelligence and Database Technologies
Edited by **Michael L. Brodie** and **John Mylopoulos**
1986/660 pages/89 illus/Cloth $38.00
Topics in Information Systems
ISBN 0-387-96382-0

Cooperative Interfaces to Information Systems
Edited by **Leonard Bolc** and **Matthias Jarke**
1986/328 pages/62 illus/Cloth $39.00
Topics in Information Systems
ISBN 0-387-16599-1

Fundamentals of Artificial Intelligence
An Advanced Course
Edited by **Wolfgang Bibel** and **Philippe Jorrand**
1986/313 pages/Paper $24.80
Lecture Notes in Computer Science, Vol. 232
ISBN 0-387-16782-X

For information on these and other Springer-Verlag titles, visit booth #203 at AAAI '87 in Seattle, or write to: Springer-Verlag New York Inc., Attn: G. Kiely, 175 Fifth Avenue, New York, NY 10010. To order by credit card, call TOLL FREE 1-800-526-7254 (in NJ, 201-348-4033).

Springer-Verlag
New York Berlin Heidelberg
London Paris Tokyo

CIRCLE 75 ON READER SERVICE CARD

PROLOG benchmarks

While the benchmarks used for this wrap-up are not the last word on the subject of benchmarking PROLOGs, we believe they represent a step forward in the business of comparing PROLOG implementations on some logical basis.

Benchmarking was done on a Zenith AT-compatible running at 12 MHz. Chalcedony PROLOG/i, Cogent PROLOG, and VML PROLOG had timing mechanisms that were too crude to make accurate timings and are not included in the benchmarks table.

The benchmarks are set up as two files. One file (DRIVER.PRO) contains the routines that call and manage the invocation of the predicates in the main benchmark file (BENCHES.PRO). The main benchmark file has a total of 22 benchmark calls, producing 33 benchmark output messages. The benchmarks are designed so that the overhead time for invoking the predicate is deducted from actual execution time. Each benchmark was run for 200 iterations, except where otherwise noted in the table:

■ *Tail_call_atom_atom* tests the response in executing 100 determinate tail calls of the form:

```
p1(a) :- p2(a).
p2(a) :- p3(a).
p3(a) :- p4(a).
```

and so on to the last two lines:

```
p99(a) :- p100(a).
p100(a).
```

■ *Binary_call_atom_atom* mixes 63 determinate nontail and 64 determinate tail calls. The former calls look like this:

```
q1(a) :- q2(a),q3(a).
q2(a) :- q4(a),q5(a).
```

The determinate tail calls were simple PROLOG facts:

```
q76(a).
q77(a).
```

continuing to 127 total calls.

■ *Cons_list* constructs a 100-element list nonrecursively. The calls look like this:

```
r1([1|R]) :- r2(R).
r2([2|R]) :- r3(R).
```

and so on to the final call of

```
r101([]).
```

■ *Walk_list* performs a nonrecursive walk down a 100-element list.

■ *Walk_list_rec* walks down the same 100-element list recursively.

■ The sixth benchmark is one of the most rigorous. It requires a recursive walk down 100, 200, 400, 800, and 1,600 copies of a 100-element list. In the tables, these timings are given under the heads of *args(1)*, *args(2)*, and so forth.

■ *Cons_term* nonrecursively creates a term with 100 nodes. The calls look like this:

```
s1(f(1,R)) :- s2(R).
s2(f(2,R)) :- s3(R).
```

and so forth until 100 nodes are constructed.

■ *Walk_term* performs a nonrecursive walk down the 100-node term constructed in *Cons_term*. *Walk_term* does the same walk recursively.

■ *Shallow_backtracking* analyzes the quality of shallow backtracking.

■ *Deep_backtracking* tests the deep backtracking speed.

For both *shallow_backtracking* and *deep_backtracking*, the benchmarks assume no indexing takes place on the second argument. Both use a consistent fact list of 100 elements, similar to:

```
b(_X, 1).
b(_X, 2).
```

and so forth.

For the shallow backtracking test, the call to this set of facts is:

```
shallow :- b(_X, 100).
```

and for the deep backtracking, it looks like this:

```
deep :- b(_X,Y), Y=100.
```

■ *Choice_point* pushs 100 choice points onto a PROLOG's stack with a simple but elegant program set:

```
c1(a) :- c2(a).
c1(a).
c2(a) :- c3(a).
c3(a).
```

■ *Trail_variables* tests the ability to manage trail variables, generally part of memory management. The benchmark creates 100 choice points and trails 100 variables with code similar to:

```
trail :- t1(X), !
t1(a) :- t2(_X).
t1(b).
t2(a) :- t3(_X).
t2(b).
```

This process repeats until 100 points and trail variables are generated.

■ *Medium_unify* and *deep_unify* test the unification of terms that are spatially small but textually quite large.

■ *Integer_add* performs 100 integer additions nonrecursively. The benchmark avoids calls that would give compilers with certain optimizations an unfair advantage. Here is a sample line from the predicate list:

```
a1(M, K, P) :- N is M + K,
      a2(N, 2, P).
```

■ *Floating_add* is essentially the same as *integer_add* but uses floating-point math instead.
■ The 17th benchmark tests the integrity of the benchmarks with 100 calls to an argument located at the first, second, fourth, eighth, and 16th positions in a complex term. If the benchmarks are running properly, timings should vary little (if at all) across positions within an implementation. Results are listed under *arg(1), arg(2),* and so forth.
■ *Index* performs 100 indexed calls, indexing on the first argument. The first few lines of code look like:

```
ix(1) :- ix(10000).
ix(4).
ix(9) :- ix(4).
ix(16) :- ix(9).
```

■ *Assert_unit* asserts 1,000 unit clauses. *Access_unit* then accesses 100 dynamically created clauses with the first argument instantiated.
■ *Slow_access_unit* does the same thing as *access_unit* but with the second argument instantiated, a substantially slower process.
■ The last benchmark tests the quality of the implementation of *setof* and *bagof* with the special test *double_bagof*. Here's the code for these tests:

```
bench_mark(setof, 10, setof
      (X, Y^pr(X, Y), S),
   dummy(X, Y^pr(X, Y), S)).
bench_mark(bagof, 10, bagof
      (X, Y^pr(X, Y), S),
   dummy(X, Y^pr(X, Y), S)).
bench_mark(double_setof, 10,
   setof((X,S), setof(Y, pr(X, Y), S), T),
   dummy(S, setof(Y, pr(X, Y), S), T)).
```

The benchmarks for this wrap-up were written primarily by Fernando C.N. Pereira, Ph.D., of SRI International in Menlo Park, Calif. Having worked with PROLOG since 1975, Pereira is a seasoned PROLOG designer, implementer, and programmer.

The concepts behind these benchmarks are the product of many peoples' work and views. Pereira wishes to acknowledge the contributions of Richard O'Keefe, Paul Wilk, and David Warren to the field of PROLOG benchmarking. The complete benchmark suite is available from the *COMPUTER LANGUAGE* Bulletin Board Service and CompuServe Forum.

By Dan Shafer

ARE YOU STUCK?

Stuck with PL/I applications running only on mainframes or minis?

Language Processors, Inc., an innovator in compiler technology, has your solution: a PL/I compiler that rescues your applications from aging mainframes and from proprietary architecture minis.

LPI-PL/I allows you to retain your present PL/I investment by letting you port the applications to just about any UNIX or XENIX-based computer.

LPI-PL/I is a true compiler that produces fast and compact machine code. You can select from several levels of optimization (local, global or machine dependent) to produce efficient applications. And our LPI-DEBUG lets you test and debug programs in PL/I, not machine language.

LPI-PL/I is just one of an entire family of compilers. The LPI family allows you to develop every subprogram of an application in the LPI language best suited for the job, and then execute cross-language calls between the subprograms. You choose the language – we will supply the compiler – with the support that made us stand above the crowd.

Copyright 1987 by Language Processors, Inc. LPI and LPI Logo are trademarks of Language Processors, Inc. The companies mentioned above own numerous registered trademarks.

And next time you buy a computer, you won't be stuck – LPI languages will be on it.

LPI-PL/I is a full implementation of ANSI PL/I X3.74-1981 General Purpose Subset. It is compatible with DG and Prime implementations of PL/I and has extensions for compatibility with mainframe and DEC implementations. Some of these extensions are: SELECT, BYTE, LEAVE, %REPLACE.

Available on MC680X0-based computers, such as Altos, Apollo, Arete, AT&T, CT, NCR, Sun and Unisys.
NEW: on AT&T 3B2, 3B5 and 3B15 Series computers.
NEW: on Intel 80386-based computers.

Call the LPI Retail Group now to order, or send the coupon. Prices start at $995 for 80386 machines.

I need more information on LPI-PL/I for the following computers:

I need more information on your other language(s):

Name Title (or attach your business card)

Address Phone

LPI
The Language Source

Language Processors, Inc.
400-1 Totten Pond Road
Waltham, MA 02154
Telephone: 617-890-1155
Telefax: 617-890-5633
Telex: 951-671

CL 7/87

LPI: PL/I, COBOL, RPG II, BASIC, FORTRAN, PASCAL, C, DEBUG

CIRCLE 8 ON READER SERVICE CARD

Benchmarks (milliseconds)

Manufacturer and product	Tail_call_atom_atom	Binary_call_atom_atom	Cons_list	Walk_list	Walk_list_rec	Args (1)	Args (2)	Args (4)	Args (8)	Args (16)	Cons_term	Walk_term	Walk_term_rec	Shallow_back_tracking	Deep_backtracking
Applied Logic Systems ALS PROLOG	1.90	3.50	3.30	2.75	2.50	2.75	4.40	7.65	14.85	—[1]	3.25	3.00	3.05	3.05	9.60
Arity Arity/PROLOG	69.55	399.05	957.00	971.35	75.80	74.90	88.60	112.60	166.40	270.50	70.60	75.50	73.90	56.90	130.90
Logicware MPROLOG	21.0	26.0	25.0	31.0	46.0	47.0	58.0	81.0	127.0	218.0	25.0	31.0	—[2]	16.0	62.0
Expert Systems International PROLOG-2 Professional	23.0	31.3	37.6	37.6	51.9	52.4	68.9	101.0	163.9	290.8	33.5	37.3	51.9	17.6	51.9
Borland International Turbo PROLOG	1.50	1.70	3.40	not run	not run	not run	not run	not run	not run	not run	not run	not run	not run	184.00	23.50

Manufacturer and product	Choice_point	Trail_variables	Medium_unify	Deep_unify	Integer_add	Floating_add	Arg (1)	Arg (2)	Arg (4)
Applied Logic Systems ALS PROLOG	5.20	6.05	3.05	190.70	8.47	164.90	—[1]	—[1]	—[1]
Arity Arity/PROLOG	380.60	65.90	2.70	190.60	176.20	459.79	159.80	159.85	159.90
Logicware MPROLOG	25.0	31.0	7.0	468.0	116.0	—[3]	159.0	159.0	159.0
Expert Systems International PROLOG-2 Professional	27.2	25.0	05.7	381.2	109.6	118.1	94.5	97.7	104.3
Borland International Turbo PROLOG	12.00	not run	not run	not run	1.70	10.00	not run	not run	not run

Manufacturer and product	Arg (8)	Arg (16)	Index	Assert_unit	Access_unit	Slow_access_unit	Setof	Bagof	Double_bagof
Applied Logic Systems ALS PROLOG	—[1]	—[1]	6.60	2,420.00[4]	6.10	17.00	275.00	10.00	719.00
Arity Arity/PROLOG	160.10	176.90	237.00	15,930.00[4]	6.60	231.00	not run	not run	not run
Logicware MPROLOG	159.0	163.0	1,241.0	19,500.0[4]	25.0	77.0	not run	not run	not run
Expert Systems International PROLOG-2 Professional	116.7	143.1	1,552.1	9,010.0[4]	32.4	89.0	—[5]	—[5]	—[5]
Borland International Turbo PROLOG	not run	not run	10.00	610.00[4]	not run	not run	not run	not run	not run

1. Benchmark failed due to too many parameters in predicate.
2. Benchmark failed due to heap or stack overflow.
3. Benchmark failed because floating-point math was not supported.
4. Only PROLOG-2 Professional supported the full 1,000 asserts in *Assert_unit*.
5. Benchmark yielded invalid results; the test time reported was less than the control predicate time.

Table 1.

Authoritative C Expertise from Plum Hall: Training, Books, and the Plum Hall Validation Suite

Organizations seeking impartial C expertise have turned to Plum Hall since 1979. C Workshops, Introductory and Advanced, are available both in-house and on a regular public schedule. Plum Hall textbooks constitute the most comprehensive integrated series on the C Language.

The new Plum Hall Validation Suite is a large set of C programs for testing and evaluating C compilers and interpreters. The Suite provides compiler writers a cost-effective distillation of the validation and testing experience of several projects. And for the user organization confronting important compiler-product selections, the Suite is an effective tool for detailed feature-by-feature compiler comparisons.

The Plum Hall Validation Suite is tracking the development of the ANSI C Standard. Each "shall," "shall not," constraint, and syntax rule of the Standard will be tested in the Suite. Over 22,000 lines of carefully written C, plus over a million bytes of script generated C, provide comprehensive coverage.

Plum Hall at your service

For fast next-day shipment of book orders, for information about top-quality training, and for details about the Plum Hall Validation Suite, call

Joan Hall
President
Plum Hall Inc
1 Spruce Avenue
Cardiff NJ 08232
609-927-3770

CIRCLE 80 ON READER SERVICE CARD

General features

Manufacturer and product	Compiler	Built-in editor	Debugging features	Windowed environment	Graphics	Tutorial	Sample programs	On-line help	Support	Direct-access files	BIOS access	DOS function access	DOS access from development environment	Data base interface	Access to other languages
Applied Logic Systems ALS PROLOG	yes	yes	trace	no	yes	no	yes	no	phone	no	yes	no	yes	sep.	C
Arity Arity/PROLOG	yes	no	trace	no	yes	no	few	no	phone	yes	yes	yes	yes	sep.	C, FORTRAN, Pascal
Chalcedony Software PROLOG/i	no	yes	trace	no	yes	yes	few	no	users group	no	no	yes	yes	yes	C, Pascal
Cogent Software Cogent PROLOG															
Interpreter	no	no	trace	no	ST	yes	yes	no	phone	no	yes	yes	yes	yes	no
Compiler	yes	no	trace	yes	ST	yes	yes	yes	phone	yes	yes	yes	yes	yes	no
Logicware M PROLOG	no	yes	trace	no	yes	sep.	few	no	phone	yes	yes	yes	yes	no	Assembler, C, Pascal
Expert Systems International PROLOG-2 Professional	no	no	trace	yes	yes	no	yes	yes	phone	yes	yes	yes	yes	no	Assembler, C
Borland International Turbo PROLOG	yes	yes	trace	yes	yes	yes	yes	yes	phone	yes	yes	yes	yes	sep.	Assembler, C, Pascal
Auomata Design Associates VML PROLOG	no	no	trace	no	yes	brief	yes	no	phone	no	yes	yes	yes	no	no

Sep. = Separate.
ST = Simple text.

Programming support

Manufacturer and product	Stand-alone programs	Interactive execution	Supports modular programming	Clocksin and Mellish compatible	Exception handling	Predicate indexes	Source code for built-in predicates	Dynamic rules	Grammar rules	Tail recursion optimized	Garbage collection	Floating-point numbers	Minimum memory
Applied Logic Systems ALS PROLOG	yes[1]	yes	yes	yes	no	yes	yes	yes	yes	yes	yes	yes	512K
Arity Arity/PROLOG	yes	yes	no	yes	no	yes	no	yes	yes	yes	yes	yes	512K
Chalcedony Software PROLOG/i	no	yes	no	yes	no	no	no	no	no	no	yes	yes	256K
Cogent Software Cogent PROLOG													
Compiler	yes[1]	yes	no	yes	no	yes	no	yes	yes	no	yes	yes	256K
Interpreter	no	yes	no	yes	no	no	no	yes	no	no	yes[2]	no	256K
Logicware MPROLOG	yes[1]	yes	yes	yes[3]	yes	no	no	yes	yes	yes	yes	no	512K
Expert Systems International PROLOG-2 Professional	yes[1]	yes	yes	yes	yes	yes	no	yes	yes	yes	yes	yes	384K
Borland International Turbo PROLOG	yes[4]	yes	yes	no	no	no	no	no	no	yes	yes	yes	384K
Auomata Design Associates VML PROLOG	yes[1]	yes	yes	yes	limited	no	no	yes	yes	yes	yes	yes	512K

1. Programs require run-time system to be run separately.
2. Garbage collection is manually initiated.
3. Clocksin and Mellish compatible in DEC-10 mode.
4. Stand-alone programs available as .EXE files.

Table 2.

C CODE FOR THE PC

source code, of course

C Source Code

FSP (screen manager)	$400
GraphiC 4.0 (high-resolution, DISSPLA-style scientific plots in color & hardcopy)	$275
Essential C Utility Library (400 useful C functions)	$160
Essential Communications Library (C functions for RS-232-based communication systems)	$160
Panache C Program Generator (screen-based database management programs)	$150
B-Tree Library & ISAM Driver (file system utilities by Softfocus)	$100
The Profiler (program execution profile tool)	$100
QC88 C compiler (ASM output, small model, no longs, floats or bit fields, 80+ function library)	$90
CBTree (B+tree ISAM driver, multiple variable-length keys)	$80
ME (programmer's editor with C-like macro language by Magma Software)	$75
Wendin PCNX Operating System Shell	$75
Wendin PCVMS Operating System Shell	$75
Wendin Operating System Construction Kit	$75
Entelekon C Function Library (screen, graphics, keyboard, string, printer, etc.)	$70
Entelekon Power Windows (menus, overlays, messages, alarms, file handling, etc.)	$70
EZ_ASM (assembly language macros bridging C and MASM)	$60
Make (macros, all languages, built-in rules)	$50
Vector-to-Raster Conversion (stroke letters & Tektronix 4010 codes to bitmaps)	$50
Coder's Prolog (inference engine for use with C programs)	$45
PC/MPX (light-weight process manager; includes preemption and cooroutine packages)	$45
Biggerstaff's System Tools (multi-tasking window manager kit)	$40
TELE Kernel (Ken Berry's multi-tasking kernel)	$30
TELE Windows (Ken Berry's window package)	$30
Clisp (Lisp interpreter with extensive internals documentation)	$30
Translate Rules to C (YACC-like function generator for rule-based systems)	$30
6-Pack of Editors (six public domain editors for use, study & hacking)	$30
ICON (string and list processing language, Version 6 and update)	$25
LEX (lexical analyzer generator)	$25
Bison & PREP (YACC workalike parser generator & attribute grammar preprocessor)	$25
C Compiler Torture Test (checks a C compiler against K & R)	$20
A68 (68000 cross-assembler)	$20
Small-C (C subset compiler for 8080 and 8088)	$20
tiny-c (C subsubset interpreter including the tiny-c shell)	$20
Xlisp 1.5a (Lisp interpreter including tiny-Prolog in Lisp)	$20
List-Pac (C functions for lists, stacks, and queues)	$20
XLT Macro Processor (general purpose text translator)	$15
C Tools (exception macros, wc, pp, roff, grep, printf, hash, declare, banner, Pascal-to-C)	$15

Data

Webster's Second Dictionary (234,932 words)	$60
U. S. Atlas (29,000 cities with display program)	$40
KST Fonts (13,200 characters in 139 mixed fonts: specify TEX or bitmap format)	$30
The World Digitized (100,000 longitude/latitude points with display program source code)	$30
NBS Hershey Fonts (1,377 stroke characters in 14 fonts)	$15
U. S. Map (15,701 points)	$15

The Austin Code Works

11100 Leafwood Lane

Austin, Texas 78750-3409

(512) 258-0785

Free shipping on prepaid orders **MasterCard/VISA**

CIRCLE 81 ON READER SERVICE CARD

THE PROGRAMMER'S SHOP

provides complete information, advice, guarantees and every product for Microcomputer Programming.

spaghetti

BASIC

↓

indented, structured

C/TURBO

Benchmark	BASICA	Turbo	Lattice C
Sieve	595 sec	4 sec	3 sec
Eight queens	597 sec	5 sec	2 sec
Matrix (single)	50 sec	6 sec	6 sec
Matrix (double)	55 sec	No double	6 sec

BAS_C $169 **BAS_PAS $129**

Free information about BAS_C,BAS_PAS, call Gotoless Conversion, Box50068, Denton, TX76206
Phone (214) 221-0383 **Demo Disk $5.00**

TP2C™ Turbo Pascal to C Translation for POWER, PRACTICALITY, and PORTABILITY

- Write More Powerful & Flexible Programs
- Easily Get Started with C Programming
- Keep Your Valuable Pascal Investment
- Port Your Pascal Programs to UNIX and Many Other Environments

TP2C translates better than 95% of your Turbo source; the clearly written documentation contains guidelines for final hand-conversion.

TP2C produces standard C code direct from Turbo Pascal programs. Create function libraries from Pascal routines. Use ***TP2C*** as a C learning aid. Move Turbo programs to Unix or other machines. Translate whole programs or program segments. ***TP2C*** even handles nested procedures and inline assembly statements.

Try ***TP2C*** for 30 days without risk. If you're not satisfied within that time, return it for a full refund.

For PC-DOS and MS-DOS machines.

List: $249
Ours: $199

BISS of Louisiana
(504) 295-0260
5432 Superior Drive, Suite A-1, Baton Rouge, LA 70816

Add Data Security to Your C Programs

The SECURITY LIBRARY™

Add Fast or Thorough Encryption or Compression to Your Programs WITHOUT Royalties

Build the safest and most popular methods of protecting your data directly into your program without paying royalties. Don't worry if you're not sure what to use. The discussion of security methods in the manual (with demos on disk) will help you make a choice based on your application.

With Security Library you can:

- Keep files secure on a multi-user network
- Speed up data transmission and communications through data compression
- Control access based on privileges given to a hard disk or LAN user

"I create custom software for business applications. I'm using the Security Library to encrypt so that certain information will not be readily available. The documentation is excellent. It gives a thorough explanation about how you can secure a file."
— Bruce Philips, Custom Software Design
Virginia Beach, CA

You Choose the Security Level

Algorithms provided include The National Bureau of Standards' Data Encryption Standard (DES) and the Vernam and Vegenere ciphers. Encrypt a 10K file in 3 seconds with one method or 50 seconds with another.

Six algorithms are provided, along with password and non-password encryption schemes.

Valuable Extras

The Huffman coding routine can *reduce the size of a file* by 25 to 50%. The routine to change the attribute bytes of a file can make that file invulnerable to casual browsing or accidental deletion. There's even a program to *change every byte of a file to a null character* - not even un-erase programs can recover it then!

Call 800-821-2492 to order Security Library risk-free for only $125. Source & Object is $250.

Solution Systems™
335-P Washington St.,
Norwell, MA 02061
(617) 659-1571

Requires MSDOS 2.0+. Works with Microsoft C and Computer Innovations' C86. Please specify compiler when you order.

Object only: $115
With Source: $229

Full refund if not satisfied in 30 days.

Call Today for FREE detailed information or try RISK-FREE for 31 days

800-421-8006
HOURS: 8:30 A.M.-8:00 P.M. E.S.T.

5-L Pond Park Road, Hingham, MA 02043 Mass: 800-442-8070 or 617-740-2510 5/87

FREE*: Probe the Minds of 19 World-Class Programmers!

Programmers at Work is a fascinating collection of interviews with industry leaders like Jon Sachs, Wayne Ratliff, Dan Bricklin, and Bill Gates. "Prefaced by a short biography and including program doodles and samples of source code, each interview examines the forces, the events, and the personality traits that have influenced the programmers' work."

FREE* or Only $14.95! — *from the jacket*

NOW: Special OFFER!
via coupon (or mention this ad)

* Order any product on this page OR $250 of any software and get Programmers at Work **FREE!**

Payment method: ___ Check/MO ___ MasterCard/VISA
Acct. # _____ Exp.: _____
Your Computer: _____ O/S: _____
Name: _____
Street: _____
City: _____
State: _____ Zip: _____
Item: _____ Price: _____
_____ _____
UPS Ground $3/item, Air $6/item. (except book): _____
(Call about other methods) **Total:** _____
Offer expires 9/30/87.

The Programmer's Shop, 5 Pond Park Rd., Hingham, MA 02043
(800) 421-8006 or in MA (617) 740-2510 L

CIRCLE 82 ON READER SERVICE CARD

LISP PC products: Survival of the species

By Julia Hodges, Jon Roland, and Richard Parke

Some of the most creative programming done on computers today is in LISP. The variety of LISP projects is reflected in the different approaches to the language's implementation on the microcomputer.

The following are reviews of a diverse collection of microcomputer LISPs. The products covered are ExperTelligence's ExperCommon LISP, Gold Hill Computers' Golden Common LISP, Integral Quality's IQLISP, ProCode International's Waltz LISP, Soft Warehouse's muLISP-86, Sapiens Software's Star Sapphire LISP, and Solution Systems' TransLISP.

Some of the packages offer features that make them uniquely suited for particular purposes. One has a LISP-to-C compiler, which, so far as we know, is the first such product in commercial distribution. Two of the packages implement large subsets of the Common LISP standard.

Among the packages is muLISP, one of the oldest microcomputer LISP implementations. muLISP is generally accompanied by the muMATH package of math functions. For a long time, these functions offered almost the only convenient way to do symbolic math and infinite-precision arithmetic on a micro.

Not everyone agrees that LISP should be the language of choice for commercial AI development or even prototyping. For commercial products, C has emerged as the language of choice.

PROLOG is considered an AI contender, especially as more mature versions emerge. A new language, NIAL, overcomes some of the disadvantages of older languages like LISP and APL, and new versions of Forth are taking their place in some serious applications.

The winners of this debate are programmers and the users of the programs they write. Innovations in one language rapidly spread among the others. Concepts like incremental memory compilers and object-oriented programming that were once thought exotic are now accepted as useful and powerful tools for improving programmer productivity and product quality.

Whether LISP ultimately survives is perhaps less important than the innovations that are being pioneered in LISP today. In virtually every area, important work is being done in LISP. Here are some of the packages that can let you join the fray.—J.R.

ExperTelligence ExperCommon LISP

ExperTelligence's ExperCommon LISP is an implementation of a large subset of Common LISP for the Apple Macintosh. As a product for the Mac, it uses the machine's operating system and toolbox to good advantage.

Although applications can be delivered on a Macintosh with only 512K RAM, the system requires at least 1MB to run. I reviewed ExperCommon LISP on a Macintosh SE with 1MB RAM and a 20MB hard disk.

The package's two main programs fit on a single 800K disk, but there is so little space left that a hard disk is required for any real programming. The first program, ExperCommon LISP, contains the LISP kernel and creates the programming environment. The second, ECL Compiler, compiles LISP programs (called forms) into an area of memory called the listener buffer. The compiler can be loaded manually or automatically on boot-up; loading takes about 2 minutes. The main disk also has a configuration file for allocating system resources.

When the package starts up, the screen is divided into two windows. The upper half of the screen is the listener window where single-line LISP forms can be entered, compiled, and executed. The lower half is the edit window where multi-line LISP forms can be entered and edited but not executed. Both windows can be resized with the mouse.

LISP forms are written by entering the name of the compiled form and any necessary arguments in the listener window. By default, the results of execution are displayed in the listener window below the program line.

ExperCommon LISP differs from most microcomputer LISPs in several ways. First, there is no file interpreter. Most LISP packages consist of a file interpreter and perhaps a file compiler. ExperCommon LISP has a file compiler available as an option, but the main compiler is an interactive incremental memory compiler.

The incremental memory compiler compiles source code from one or more edit buffers into 68000 native code. The code resides in the listener buffer and can be accessed through its window. In this, ExperCommon LISP is similar to Texas Instruments' PC Scheme, Q'NIAL, and most versions of Forth and APL, which use an area of memory to hold the results of compilation and program execution.

You can compile all of your forms to the active edit buffer or highlight one or more with a mouse and just compile those. Forward referencing is permitted within a group of forms. However, before referencing a user-defined form you must have previously compiled it during the session. Any forms that have been compiled can be referenced; forms can be recompiled independently. One feature lacking is the ability to save the listener buffer to a file with all forms and variable bindings intact, a feature that would be helpful during development.

The edit window's editor blinks matching parentheses. Using the blinking parentheses and a mouse, you can mark a block of text for copying, moving, or deleting. Checking for unpaired parentheses is more difficult, requiring you to place the cursor on each parenthesis and then scan for a matching one blinking somewhere else.

Since ExperCommon LISP doesn't support paging or overlaying large programs, available memory limits program size. The current version of ExperCommon LISP can use up to 4MB. The version for the Macintosh II is expected to support more memory and the complete Common LISP standard.

The file compiler is distributed as a separate option. With the ResEdit facility, you can use the file compiler to create stand-alone programs. The stand-alone programs contain a copy of the ExperCommon LISP kernel but don't require the large amount of memory used by the

LISP environment. The minimum size of stand-alone programs is about 100K; they can be distributed without royalties.

You have to write your own user interface, but facilities are provided to use the entire Macintosh Toolbox. A separate product, the ExperInterface Builder, can be used to interactively generate interfaces without familiarity with the Macintosh Toolbox internals.

Almost all of the often-used Common LISP features are implemented; multidimensional arrays and handles are supported. The only features I missed were the *BIGNUM* data type and support of the *INITIAL-ELEMENT* keyword by the *MAKE-ARRAY* function.

ExperCommon LISP goes beyond the Common LISP standard by supporting object-oriented programming. The package's support goes beyond that of LOOPS and Flavors, more closely resembling Smalltalk-80. Over 600 classes support the more than 700 Macintosh Toolbox functions. However, ExperCommon LISP does not support multiple inheritance, only linear inheritance, but multiple inheritance is seldom used by most LISP programmers.

The Gabriel benchmarks I ran gave acceptable results, even though I didn't edit the configure file to optimize the use of available memory. More than half of the compilation and execution time was taken up by garbage collection. With more memory and an edited configure file to take advantage of it, compile and execution speeds could probably be reduced by a factor of three. ExperTelligence claims that v. 2.2's garbage collector is four times faster than v. 2.1's.

A more interesting claim I couldn't confirm is that ExperCommon LISP on the Macintosh II often outperforms Common LISP on the Symbolics 3600. For floating-point operations, ExperCommon LISP is said to be faster by a factor of five since it makes use of the 68881 floating-point processor, which is not available on the Symbolics 3600. If true, this level of performance could make the Macintosh II the system of choice for both research and commercial use.

ExperCommon LISP's documentation is excellent. It does not include Guy Steele's *Common LISP: The Language* (MIT Press, 1984), but the main package's two manuals and a third manual that supports Macintosh Toolbox development are well suited for learning Common LISP or application development on the Macintosh.

I'd like to see more sample programs. Public-domain code is available, but it will need to be rewritten to some extent. Examples of converting between versions of LISP would be useful. Also, while ExperCommon LISP's debugging facilities are excellent, the documentation does not go into sufficient detail on interpreting or responding to some of the error messages.

ExperCommon LISP is an excellent package for learning Common LISP and developing serious applications. It implements most of Common LISP and provides satisfactory performance on a 1MB Macintosh, but for full Common LISP or more serious applications I recommend a Macintosh II with plenty of RAM. In addition, ExperCommon LISP provides a good introduction to the Macintosh Toolbox and object-oriented programming.—J.R.

Gold Hill Computers
Golden Common LISP

Gold Hill's Golden Common LISP (GCLISP) is a Common LISP programming environment for the IBM PC and 100% compatibles. GCLISP is designed for use by both novice and experienced LISP programmers.

The package includes the GCLISP interpreter, which implements a major subset of Common LISP; the San Marco LISP explorer, an on-line interactive tutorial; and GMACS, a full-screen text editor with debugging facilities that's modeled along the lines of the EMACS editor.

GCLISP provides on-line help facilities for every function and variable. Two books are also included with the package: the second edition of *LISP* by Patrick Winston and Berthold K.P. Horn (Addison-Wesley, 1984) and *Common LISP Reference Manual* by Guy L. Steele Jr. (Digital Press, 1984).

GCLISP requires 1MB of memory. At least this much memory must be available for GCLISP to be installed on a hard disk.

The interactive tutorial is excellent. While it is far too detailed for the experienced LISP programmer, the simple explanations, many examples, and use of graphics make it an excellent teaching tool. You don't have to go through the entire tutorial; provision is made for jumping around. Your progress through the tutorial is logged when you exit.

The debugging facilities are also excellent. On-line help allows the programmer to list the arguments required for a given function, check the meanings of special keys, and refer to the documentation on various functions.

The package includes the usual *break, step,* and *trace* found in most Common LISPs. If you use *break* or if an error occurs during a function's execution, a new listener level is created. At the new level, you have access to all the services available at the top level. You can return from this new level with the function *clean-up-error* or you can try to track down the bug with the *backtrace* function, which lists all GCLISP forms that have not yet finished.

The implementation is a substantial subset of Common LISP. The only Common LISP feature notably missing is the *eval-when* function, which allows pieces of code to be executed at compile time only, load time, or when code is interpreted. The function's absence doesn't pose a significant problem since its use is quite limited and has few practical applications.

GCLISP evaluates functions as soon as the last right parenthesis is typed—there's no need to press the Enter key. Execution speed for functions is quite impressive.

This is an excellent package for both novice and veteran LISP programmers. The on-line tutorial will be appreciated by the beginner. The speed of execution, excellent debugging facilities, and Common LISP facilities make GCLISP a good choice for anyone shopping in the LISP market.—J.H.

Integral Quality
IQLISP

Integral Quality's IQLISP is designed to take full advantage of the IBM PC's features for the development of complex systems. With its speedy development, testing, and debugging of programs, IQLISP is well suited for applications that must be built quickly.

In general, IQLISP provides a good LISP environment. Once you get accustomed to how the system operates, program development is a very comfortable process. A variety of I/O functions are included, among them formatted I/O, command-line arguments, and windowing capabilities.

Many of the DOS functions (*DIR, CHDIR, MKDIR,* and the like) can be accessed from IQLISP. For scientific applications, IQLISP provides most of the commonly used mathematics functions, such as *SQRT, SIN, COS, ABS,* and *ATAN,* as well as math coprocessor support.

The package includes an editor that can be used within the LISP environment, which can be quite convenient during program development. However, the editor is a command-driven, object-oriented LISP editor and somewhat difficult to use due to a rather limited set of commands.

For debugging, a diagnostic package includes the usual LISP facilities like *trace* and *break*. An error monitor allows easy examination of the stack when an error condition occurs. This utility is similar to GCLISP's backtrace utility.

A highly unusual feature of IQLISP is the inclusion of structured programming macros. These macros (*FOR, REPEAT,* and *LOOP*) make it easy to use iteration

386•386•386•386•386•386•386•386•386•386•386•386•386•386•386•386•386•

NOW!

386
C *and* Pascal
for MS-DOS

MetaWare Incorporated announces the *first* available C and Pascal compilers that generate

protected-mode 80386 code

for running on any 80386 machine that runs MS-DOS (e.g., the Compaq Deskpro 386 or the IBM Personal System/2 Model 80). The compilers are functionally identical to our well-respected 8086/286 MS-DOS **High C** ™ and **Professional Pascal** ™ compilers, but now you can get them generating 80386 code.

There's no reason to wait! Industry leaders such as ANSA and Fox Software are already converting their 16-bit database products to 32-bit protected mode, getting increases in speed and functionality. Don't wait years for Microsoft's 386DOS — your competition will have a big jump on you!

Expand your application to the large 32-bit address space and the full 32-bit registers of the 80386. Contact MetaWare for your 80386 software solution today!

(408) 429-6382, telex 493-0879.

MetaWare™

INCORPORATED

903 Pacific Avenue, Suite 201 • Santa Cruz, CA 95060-4429

CL 07/87

CIRCLE 83 ON READER SERVICE CARD

Bpcs
Breakpoint Computer Systems, Inc.

Introducing: QMan, a true relational data base management system which supports both interactive and imbedded queries. Based on the IDL query language from Britton Lee Inc. A great foundation for applications, and an incredible package in its own right. Features include: multi user access, B+ Tree Indices, imbedded queries, sorting, and sort merge. Portable to most SYS V machines. Include machine name when ordering. **Just $895.**

QMan now available for DOS: $219

Also available through Bpcs: from Microport Systems System V/AT: UNIX™ for your AT!

	List	Bpcs
RunTime System	$199	**$179**
Development System	$249	**$225**
Text Processing System	$199	**$179**
Complete System V.2	$549	**$485**
Unlimited Upgrade	$249	**$225**

Breakpoint Computer Systems, Inc.
6701 Seybold Road, Suite 204
Madison, WI 53719

VISA and Mastercard accepted. For Microport products add $6.00 per single package or $14.00 complete system.
UNIX is a registered TM of AT&T

Call collect (608) 271-2171

CIRCLE 76 ON READER SERVICE CARD

COGENT PROLOG COMPILER

Edinburgh Prolog language, **Compiled** speed, **Interpreted** flexibilty in a full **Development Environment** for IBM PC, PC-AT.

- o Over 150 predicates (including univ, call, arg etc.)
- o Floating point, strings and database types. DCGs.
- o Complete file I/O. Total screen control. Dos and Bios calls.
- o Programmable Window System. Window Based Debugger.
- o Compiler, Interpreter, Online Help, Error Handling.
- o Garbage Collection, Dynamic Code Loader, Linker.
- o Generate and distribute applications - no royalties.
- o Needs 384K, DOS 2.0 or later.

Many documented sample Prolog programs. User and Reference manuals. All for $200 !

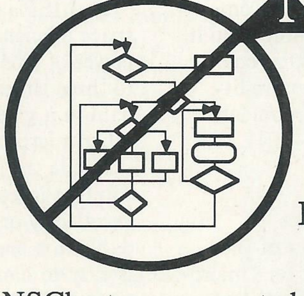
o Learn Prolog with the standalone Cogent Prolog Interpreter - Only $79

o Prolog Compiler ToolBox - Interfaces, Expert System Shells, ++ ! Only $79

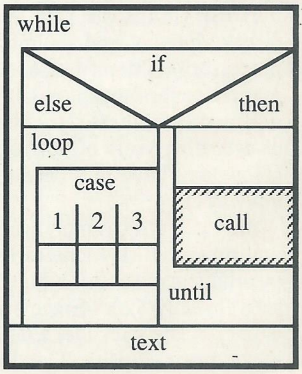
Cogent Software, Ltd.
21 William J. Hts., Framingham, MA 01701
(617)-875-6553
(Please add $5 Shipping. Mass. residents add 5% sales tax).

CIRCLE 79 ON READER SERVICE CARD

NSChart



Replace unstructured flow charts with
Nassi-Shneiderman
STRUCTURED FLOW DIAGRAMS



NSCharts are created directly from text files.

Text files consist of NSPDL, a Siltronix structured design language.

Translators to NSPDL from FORTRAN 77, C, Pascal, SDDL PDL and Caine-Farber-Gordon PDL are also available.

NSChart runs on Macintosh™ and IBM™ PCs. Both feature a What-You-See-Is-What-You-Get-Interface.

from SILTRONIX

P.O.Box 82544 San Diego, CA. 92138

Sales (800) 637-4888 Tech. Support

IBM PC is a trademark of International Business Machines Corporation
Macintosh is a trademark of Apple Computer Inc.

CIRCLE 86 ON READER SERVICE CARD

when recursion isn't practical.

An on-line help facility documents the use of the various LISP primitives. Basically, the help facility only provides syntactic information about various functions. For example, *(HELP ASSOC)* causes the system to respond with *(ASSOC <ITEM> <LIST>)*.

Surprisingly, IQLISP is case sensitive to uppercase. This makes it awkward for those who've used Franz LISP, which is case sensitive to lowercase. This design is an unnecessary inconvenience.

The documentation is quite poor. Information isn't arranged to be easily available—the major problem is the indexes. There are indexes at the end of selected chapters, but there's no comprehensive index. To find an item, you must make an educated guess as to which chapter contains the information you need and then check that chapter's index (if indeed it has an index). A comprehensive index should be added; as presently organized, the manual is entirely too difficult to use.

The manual also lacks documentation on installing the system. The system requirements section has one paragraph that instructs you to insert the IQLISP disk containing the IQ.EXE file and type *IQ*. There is no further documentation. Users of hard disk systems are left to their own devices in determining what actions are appropriate. Although loading the system is not difficult, the lack of documentation is a major oversight.

IQLISP makes no claims about being compatible with either Franz or Common LISP. Once you're accustomed to it, IQLISP provides a good working environment, but the lack of compatibility with better known LISP environments detracts from its desirability.—J.H.

Soft Warehouse
muLISP-86

muLISP (mu is the 12th letter of the Greek alphabet and a symbol for "microprocessor") is described as a high-performance, general-purpose LISP system for personal computers. It is the culmination of over nine years of development effort. (How long have PCs been around, anyway?)

muLISP is extremely well packaged and presented. It includes features found in products with much higher prices but sells for only $250. Since a good tutorial is part of the package, LISP knowledge is not a prerequisite. The documentation includes references not only to AI books and periodicals but professional societies as well. muLISP is known for its speed—an important, albeit not all-important, consideration for potential PC LISP purchasers.

According to Soft Warehouse, muLISP provides "over 400 functions, special forms, macros, and control variables. Over 260...are in machine language for maximum efficiency." Infinite-precision integers and adjustable-precision rational arithmetic are supported. The two-pass garbage collector typically takes 2 seconds to complete. Dynamic memory reallocation is done within the 512K limit.

muLISP's kernel is small; 128K is the minimum RAM required. Extremely fast execution speeds have been achieved through shallow variable binding, address-typed data structures, and a closed pointer universe.

For your money, you get a lean, mean LISP interpreter, a debugger with trace and statistics, an editor, an optional compiler, demonstration programs, a tutorial, and utility libraries for Common LISP and InterLISP functions. *CATCH, THROW, UNWIND, PROTECT,* and sequential or random file access are notable additional features.

As for performance, the product installs quickly and initializes with blazing speed. I never broke an advertised feature; the whole product-pounding process was characterized by a growing appreciation of muLISP's speed. I was tempted to bring it up on my portable PC (which runs Microsoft's Windows at glacial speed) just to gloat at how muLISP can scream even on an antique. On a Compaq 286, it breezed through the Towers of Hanoi puzzle, even with 13 rings.

muLISP's other big strength is its space efficiency. The package has the power of Common LISP and stores two to three times as much program and data in a given amount of memory as other microcomputer LISPs.

While the documentation is a little wanting in color and visual variety, it is excellently organized and eminently readable in a small, convenient binder that is easy to handle. The primitive definitions include examples, a disturbingly frequent omission from many language products.

On the downside, muLISP uses only 512K of memory. Since DOS can address 640K, and serious artificial intelligence programming relies upon massive data stores, I wonder why Soft Warehouse stopped at 512K.

I was impressed with muLISP as a conservative, disciplined LISP for the PC. It is quite fast. Soft Warehouse's advertised benchmarks are, of course, unabashedly slanted toward its products with programs that don't ask for too much memory. A beginning LISP user should have no trouble learning and using muLISP, and the experienced user will take some time to outgrow the package.

I unhesitatingly recommend muLISP-86 as a sturdy, efficient interpreter/compiler. muLISP-86's successor, muLISP-87, is now available and is said to support more functions and forms as well as object-oriented programming with a Flavors package.—R.P.

Sapiens Software
Star Sapphire LISP

Sapiens Software's Star Sapphire LISP offers two unique features: a LISP-to-C compiler and a virtual 8MB address space. In addition, Sapiens advertises access to all Common LISP functions.

Using Star Sapphire definitely requires knowledge of LISP. Unlike other micro LISPs, Star Sapphire does not have a tutorial. Instead, it is designed for the experienced programmer, especially with its C support. The programmers guide displays an unabashed attitude of respect for the user as a serious programmer willing to dive into the product and grapple with issues of LISP and C internals.

Star Sapphire offers the first LISP-to-C compiler on the IBM PC. Many major software vendors are joining the performance-minded crowd that is reimplementing in C, among them SAS Institute and Teknowledge. A typical AI application is prototyped and developed in LISP, then the lean, stripped run-time package is marketed in C. Star Sapphire astutely follows that trend.

The virtual memory design acknowledges that any truly smart system will likely need of lots of code and data in memory rather than clever, tight algorithms. This is especially true for LISP with its demons, blackboards, semantic networks, and knowledge bases. In my opinion, any LISP that doesn't support lots of memory is suspiciously toylike.

I brought up Star Sapphire with Borland International's Sidekick resident and was startled to see the message "virtual memory error #7921 : xxx near line 0." After I kicked out Sidekick, Star Sapphire came up fine; it does take a good 20 seconds for the run-time environment to come up. The run-time development system affords "the capability of building interpreters with a smaller number of functions than a full LISP and incorporation of any C or assembly routine in the interpreter."

This is a welcome and forward-looking attitude. Coding is best done in the language most suitable to a job. Stubborn adherence to a single language is counterproductive.

Virtual memory can be configured for the number of pages and megabytes of virtual memory. By default, megabytes 0-2 are LISP *CONS* space, and 2-4 are atoms (data). Megabytes 4-8 are the virtual heap. Specific workspace addresses can be accessed with the *VSTORE* and *VFETCH* functions.

The disk label bears the trademarked

legend "A Common LISP compiler for the IBM PC." The package's developers are clearly ambitious and very serious. Offering a virtual 8MB, a run-time environment, and a LISP-to-C compiler are signs of a company with big plans.

However, the product is on shaky ground with its documentation. The documentation is poorly thought out with regard to getting you the information you need, and its coverage of many issues is less than thorough. For example, the section on nonrecoverable LISP errors says that the fatal error function is "vectored to" as a result of "munged virtual strings" and "impossibly incorrect input."

For the review, I received a programmers guide to Star Sapphire, which was run off on a dot-matrix printer and stapled. The text was not paginated, was too chatty for my taste, and had several spelling errors. Sapiens states that a functional specification of Star Sapphire is available; it is deferred to as the exhaustive reference.

This is an intelligently constructed product that is still growing. For the price, its unique features keep Star Sapphire LISP among the contenders. The product suffers in delivery and appearance, but its strong promise may make up for that.—R.P.

Solution Systems TransLISP

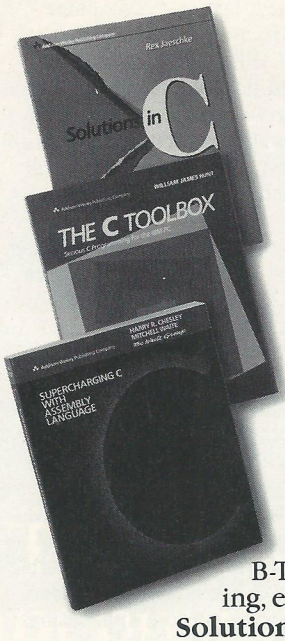
David Betz's XLISP floated around the public domain for a while as a small LISP with object-oriented programming support. Solution Systems' TransLISP is kind of a grown-up XLISP. The system is for the nonsophisticate who wants to really enjoy the product on his or her PC, and that's a breath of fresh air.

TransLISP is a nice, solid Common LISP-compatible interpreter for the novice LISP programmer. No more knowledge of LISP is assumed than can be garnered by reading the included tutorial.

The package has lots of demo programs that are designed to be both enjoyable and instructional. They serve nicely as a coding point of departure for the novice programmer who wants to see how things are done by examining a working program.

I had fun interacting with the Job Counselor program, drew some graphic shapes, and invoked the tiny PROLOG interpreter. Solution Systems wants you to enjoy its product, an attitude that isn't everything but is rare enough these days.

TransLISP's editor can be invoked from within or outside of the interpreter. Of course, it's best to edit while in the interpreter where you can execute what you just edited. To use the editor, be prepared to learn a fairly large set of special keys, key sequences, and key chords. The editor can even be used on non-IBM-compatibles.



Books to help you C better.

Supercharging C with Assembly Language by Harry R. Chesley and Mitchell Waite gives programmers familiar with C and the IBM PC, tips on creating faster, more powerful and more responsive C programs by combining its versatility with that of assembly language routines. The book includes techniques, examples and programs, and topics from how to encrypt and decrypt files to how to display fractals on the EGA.

The C Toolbox by William James Hunt offers experienced programmers valuable C tools and complete programs for creating B-Trees, terminal emulation, sorting and merging, error handling, and file I/O and bit operations.

Solutions in C by Rex Jaeschke covers problems faced daily by professional C programmers, such as memory management, optimizing program speed, and characteristics of C on UNIX and MS-DOS and PC-DOS microcomputers.

Available now wherever computer books are sold.

ADDISON-WESLEY
Reading, Massachusetts; Don Mills, Ontario

CIRCLE 87 ON READER SERVICE CARD

Personal REXX
for the IBM PC

- ★ Interpreter for the full **REXX** language, including all of the standard **REXX** instructions, operators, and built-in functions
- ★ Sophisticated string manipulation capabilities
- ★ Unlimited precision arithmetic
- ★ Direct execution of DOS commands from **REXX** programs
- ★ Built-in functions for DOS file I/O, directory access, screen and keyboard communication, and many other PC services
- ★ Compatible with VM/CMS version of **REXX**
- ★ Uses include:
  - — Command programming language for DOS
  - — Macro language for the **KEDIT** text editor
  - — Can be interfaced by application developers with other DOS applications, written in almost any language

Mansfield Software Group, Inc.
P.O. Box 532
Storrs, CT 06268
(203) 429-8402

$125 plus $3 shipping
MC, VISA, AMEX, COD, PO, CHECK

CIRCLE 71 ON READER SERVICE CARD

The product installs quickly and comes up fast. As long as I stuck to the book, the system never bombed. The manual gives a 10-page, point-by-point comparison of TransLISP to Guy Steele's Common LISP, even providing section numbers from the Common LISP book. There are quite a few "not supported" disclaimers, but this product isn't billed as a big-time LISP interpreter, after all. I didn't find any incompatibilities between TransLISP and Common LISP, but I didn't look very hard.

TransLISP's main strength is its accessibility to the novice. The documentation is delightful and includes an exhaustive index, a LISP-C-Pascal-BASIC language comparison, and a LISP bibliography. I was pleased to see an interpreter for a heavyweight AI language that's targeted at the growing number of PC owners who need a starting point somewhere and for whom confidence and comfort are essential.

Of the products I covered, TransLISP was the most fun to use and review, mainly because the book is good stuff. The package is for the novice to play with and learn LISP from on a personal computer. TransLISP begs for buyers' well-deserved attention with a Borland-like cheeriness. TransLISP was recently joined by TransLISP PLUS, which sells for $100 more.—R.P.

ProCode International Waltz LISP

ProCode International's Waltz LISP is intended to be a fast microcomputer LISP. The system comes with a full set of Common LISP functions, including such advanced flow-of-control features as *catch* and *throw*. It also has error-handling routines that allow you to interrupt a program at any time, do tracing and debugging, and then resume program execution.

I reviewed the 16-bit version of Waltz LISP; an 8-bit version that runs under CP/M-80 v. 2.0 or higher on a Z80 processor and at least 48K RAM is also available. Although both the 16-bit and 8-bit versions are capable of running in modest amounts of RAM, ProCode recommends more memory for serious development.

The package includes a full-screen editor that's WordStar-compatible. Waltz LISP's ability to handle text files and certain other features make it a suitable programming environment for the development of general-purpose applications (in addition to the AI applications normally associated with LISP). For example, Waltz LISP has a full set of string operations, including a powerful pattern-matching function.

With Waltz, you can implement random file accessing using operating system calls. This facility allows reading from and writing into a file, with the ability to access any given byte of a file. Waltz LISP also provides an interface to assembly language so that an application can conveniently combine both LISP code and assembly language routines.

Waltz LISP lacks many features normally found in Common LISP environments. For example, there is no function for defining global variables (the *defvar* function in Common LISP). Certain short forms, such as *1+* and *1-*, are also not available. Of course, the absence the short forms is not a big problem since you can easily define them for yourself.

The Waltz LISP functions for dealing with property lists of symbols are somewhat different from those of Common LISP. For example, Common LISP's *setf* function, a macro for examining and updating an access form, is missing. The *setf* macro performs the same functions as *setq, rplaca,* and *set,* all of which are included in Waltz LISP. Thus, applications written in Common LISP that make use of *setf* will have to be rewritten.

Although Waltz's *throw* and *catch* forms are syntactically the same as Common LISP's, they don't appear to work in the same way. These forms can be used to exit from a process in a nonlocal manner; their syntax is given in the Waltz LISP users manual, but there are no examples of their use.

Lack of examples is a frequent problem in the Waltz LISP users manual. For many of the functions, only the syntax is described. Of those that do have examples, some are so trivial as to be useless.

Waltz LISP's facilities for general-purpose programming add to the attractiveness of the package. Once you're accustomed to it, Waltz LISP itself is a good LISP environment. However, if you've programmed extensively in Common LISP, you may find that you have to make some fairly large changes to run existing programs.—J.H.

LISP products

ExperCommon LISP
Main package with environment and memory compiler—$995
Supplemental package with file compiler—$495
ExperTelligence Inc.
559 San Ysidro Rd.
Santa Barbara, Calif. 93108
(800) 828-0113,
in Calif. (800) 826-6144

Golden Common LISP—$495
Gold Hill Computers
163 Harvard St.
Cambridge, Mass. 02139
(800) 242-5477,
in Mass. (617) 492-2071

IQLISP—$270
Integral Quality Inc.
P.O. Box 31970
Seattle, Wash. 98103-0070
(206) 527-2918

Waltz LISP—$169
ProCode International
475 Watchung Ave.
Watchung, N.J. 07060
(800) 522-5477,
in N.J. (201) 755-5477

muLISP-86—$250
Compiler with package—$100
Purchased separately—$150
Soft Warehouse Inc.
3615 Harding Ave., Ste. 505
Honolulu, Hawaii 96816
(808) 734-5801

Star Sapphire Common LISP—$495
Sapiens Software Corp.
303 Potrero St., Ste. 59
P.O. Box 7720
Santa Cruz, Calif. 95161
(408) 458-1990

TransLISP—$95
Solution Systems
541 Main St., Ste. 410
S. Weymouth, Mass. 02190
(617) 337-6963

Julia Hodges is an assistant professor of computer science at Mississippi State University, Miss., with interests in data base systems and knowledge-based expert systems.

Richard Parke is president of Intelligence Research Group Inc. in Santa Cruz, Calif., where he does consulting, teaching, and software development. His interests include the use of microcomputers in education and small business.

Jon Roland is president of Starwood Corp., a software development firm specializing in data base applications in San Antonio, Texas. He is currently doing research in knowledge representation theory.

. . . *Software Developers* . . .

develop your software to work with most of the desktop printers on the market!

DOT DIASY LASER COLOR JET

CARDINAL POINT INCORPORATED has published (3) three volumes of the "**PROGRAMMERS' HANDBOOK OF COMPUTER PRINTER COMMANDS . . .**" Each book contains the ESCAPE and Control Code sequences for most of the personal printers made by leading manufacturers both old and new. Each book is 5½" x 8½", spiral bound, and contains an easy to use table-format. Codes are listed in text-form with hex and decimal equivalents and detailed function descriptions. The information is not duplicated from one volume to the next.

1. Vol I - '84 ——— $37.95	**4. Vol III - '86 ——— $26.95**
2. Vol II - '85 ——— $26.95	**5. 3 Vol Set ——— $77.95**
3. 2 Vol Set (I & II) — $58.95	**6. 2 Vol Set (II & III) $49.95**

(Indiana residents add 5% sales tax. Please add $3 s/h (UPS reg.) per order. Quantity Orders or Dealer Inquiries may call collect. TO ORDER CALL: (812)-876-7811, (M–F, 9-5 EST) For more detailed information, write:

Cardinal Point
INCORPORATED

P.O. Box 596-Dept-A, Ellettsville, IN 47429
We accept MC, VISA, MO, P.O.s–same day shpg.
Cks–allow 14 days. CODs– extra. UPS Blue–$5 shpg.

CIRCLE 89 ON READER SERVICE CARD

VERSION CONTROL SYSTEM

TLIB™ keeps ALL versions of your program in ONE compact library file, *even with hundreds of revisions!*

- **Fastest, most powerful** version control system you can buy. *Nothing else comes close!* TLIB updates libraries faster than some text editors can load and save files.
- *LAN-compatible!* Shared libraries with PC Net, Novell, etc. Check-in/out locking for multi-programmer projects.
- Synchronized control of multiple related source files.
- *Easy to use.* Menu or DOS command line parmeters.
- *Frugal with disk space.* Libraries are more compact than with most other version control systems. And TLIB uses no temporary files, so you can maintain a 300K library on one 360K diskette, with room to spare, even with TLIB itself on the same disk.
- Free copy of Landon Dyer's excellent public domain **MAKE** utility (.EXE, plus source code for DOS & VAX/VMS).
- **Plus:** File compare utility. Virtually unlimited source file size. Date, comments with each version. Configurable user interface, and many configurable options, like: read-only libraries; automatic tab/blank conversion; insertion of revision history comment block in the source file.

PC/MS-DOS 2.x & 3.x ***Just $99.95 + $3 s/h*** Visa/MC

BURTON SYSTEMS SOFTWARE
P. O. Box 4156, Cary, NC 27511-4156
(919) 469-3068

CIRCLE 90 ON READER SERVICE CARD

Save Time & Money
With
GeneratoR

"The Structured Program Editor and Source Code Generator that lets you design before you program"

--Saves you 30% or more in coding time....
--Create Documentation as you program....
--Computer aided software engenering helps you design and write Tight structured programs following Warnier/Orr Methodology....
--Generates Code and Psuedo Code ....
--Easy to Learn --Easy to update....
--Full Featured Text Editor....

Software Support Intro Special

Pascal Version	C Version
~~$129.95~~	~~$149.95~~
$90.99	$104.95

Call Today 913-354-1618
701 Jackson B5 Topeka KS. 66603

CIRCLE 91 ON READER SERVICE CARD

OPT-TECH SORT™

Sort/Merge/Select Program
for the IBM-PC and Compatibles
New Version 3.0 with
Btrieve & dBASE support

- High performance assembly language
- Sort multiple files on multiple keys
- Filesize limited only by disk space
- Select records on up to 10 keys
- Output records can be reformatted
- Supports many filetypes and datatypes
- Can be CALLed or used stand-alone
- $149 — Visa, MC, Amex, COD, or PO

Quantity discounts available.

To order or to receive more information call or write:

Opt-Tech Data Processing
P.O. Box 678, Zephyr Cove, NV 89448
(702) 588-3737

CIRCLE 92 ON READER SERVICE CARD

TURBOLINK+

Now Add TURBO Prolog to your Pascal Code!

"TURBO Pascal hackers will like this program."
Jerry Pournelle
Byte Magazine

- Add 512K of TURBO Pascal* to your code
- Call up to 8 memory resident TURBO Pascal modules from programs written in:
  – BASIC, BASICA, Compiled BASIC
  – MS Pascal, C, MS Fortran
  – Standard, 8087 and BCD TURBO Pascal
  – TURBO Prolog
- Add TURBO Pascal's 8087, BCD and graphic capabilities to other languages
- Automatically generate TURBO Pascal compatible inline machine code
- Use all TURBO Pascal variants in a single program
- Add AI capability to your Pascal programs
- For IBM PC and compatible computers

$69.95 Foreign orders add $10.00

S&H included
VISA/MC/Information: **(303) 971-0729**
PATHFINDER SOFTWARE, INC
P.O. BOX 43, LITTLETON, CO 80160

*TURBO Pascal and TURBO Prolog are registered trademarks of Borland International

CIRCLE 93 ON READER SERVICE CARD

STRUCTURED™
INFORMATION SYSTEMS, INC.

FINALLY . . . ASSEMBLY LANGUAGE CONTROL WITH COMPILER CONVENIENCE!

- ASSEMBLY LEVEL LANGUAGE WITH SYNTAX SIMILAR TO "C"
- HIGH LEVEL CONTROL CONSTRUCTS INCLUDING: IF-ELSE-WHILE-SWITCH-DO-WHILE
- INCREASES PRODUCTIVITY
- FULL MACRO PREPROCESSOR
- GENERATES INTEL OR MICROSOFT ASSEMBLY LANGUAGE
- RUNS UNDER PC/MS DOS 2.x AND 3.x 256K
- AVAILABLE FOR 8086/186, 8051, OR 8096
- $99 CHECK, MONEY ORDER, VISA, MASTERCARD

TO ORDER, CALL OR WRITE ™

STRUCTURED
INFORMATION SYSTEMS, INC.
86 BROAD ST., EATONTOWN, NJ 07724
OUT OF NEW JERSEY: 1-800-445-1589
IN NEW JERSEY: 1-201-544-8388
N.J. Resident Add 6% Sales Tax

CIRCLE 94 ON READER SERVICE CARD

Advertise in the COMPUTER LANGUAGE Classified Connection

This exciting new advertising section debuts in October. The classified format provides an easy, cost-effective approach to reaching technical buyers.

For more information, contact:
Linda Pizotti
(617) 232-5470

FORTRAN
ADDENDA® v5.1

Comprehensive and easy to program I/O and graphics libraries. User I/O editing with context & type checking with Helps. Or enhance your graphics with TILEng. Tools for complete template systems and device independent and windowed graphics.

ASMUTIL2®	BUTILE®
• Full/FAST Graphics	• Natural I/O System
• Full PC, AT+ Keyboards	• Windows & Pull Downs
• Full Paths/Directories	• User Input Editing
• Music, Clocks, Strings	• I/O Error Recoveries
• RS-232s, CRTs, Printers	• No I/O Overflows

- ALL MDA/CGA/EGA/Hercules modes.
- IBM, Lahey, MS(3,4) and R-M Compilers. No fees.
- Extensive and easy to read manuals and samples.
- Argument checks, defaults and toggles to speed dev.

ASMULTIL2v5.1 $95; both $165. Updates to v5.1 $45.

IMPULSE ENGINEERING

Impulse Engineering, Ray Strong • (415) 788-4611
P.O. Box 3540 • San Francisco, CA 94119-3540

CIRCLE 95 ON READER SERVICE CARD

Introductory Offer: $89.50 prepaid, Visa, MC, COD

dNPL® / Reporter

A Fourth Generation *Relational* Report Writing System for dBase files with its own full screen editor
dNPL is the *Non-Programmer's Language* for the Business Professional

- *Tabular and matrix reports, mailing labels, forms, letters*
  Interactive query for testing data and report options
  Simple, English-like, SQL-like sentences
- *Records selected by multi-function criteria, with values, ranges, strings, wild cards*
- *Up to 10 nested grouping levels with subtotals*
  Summaries with row and column totals, average, min, max, avg-sum-of-the-squares
  Cross-footing with computed totals
- *Combined sort keys up to 512 characters*
  Very fast RAM sort, with disk merge for largest files
  Matrix reports sorted across page by one variable, down page by others
- *Up to 255 memory variables*
  Up to 255 database & defined variables
  Trigonometric, date, string, sqr, sqrt, ln, exp functions
  Unlimited math expressions & nested If...Then...Else logic
  Comparisions between successive records
- *Link up to 8 databases - dBase II, dBase III, ascii files*

Write your first report within minutes

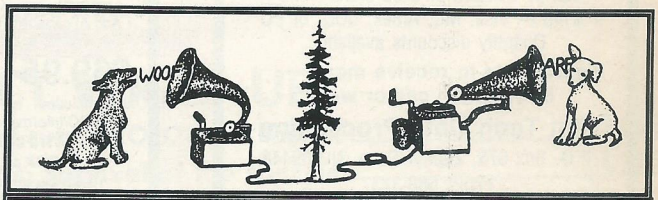
DATABASE APPLICATIONS, INC.
400 Wall Street • Princeton, NJ 08540 • 609-924-2900

CIRCLE 96 ON READER SERVICE CARD

ADVERTISER INDEX

	PAGE NO.	CIRCLE NO.
Addison-Wesley Publishing Co.	141	87
Addison-Wesley Publishing Co.	60,61	31
Aker Corp.	77	36
Applied Logic Systems	101	49
Arity	74	35
Aspen Scientific	89	88
The Austin Code Works	135	81
Bay Soft	21	13
BC Associates	119	63
Blaise Computing	5	3
Blaise Computing	11	6
Borland International	1	1
Borland International	Cover II	—
Breakpoint Computer Systems	139	76
Burton Systems Software	143	90
C Ware/DeSmet	41	—
Cardinal Point Inc.	143	89
Cogent Software Limited	139	79
CompuServe	106	41
Crackerjack Microsoftware Corp.	123	66
Creative Programming	18	—
Custom Software Systems	21	14
Database Applications	143	96
Datalight	8	4
Desktop AI	117	77
Digitalk Incorprated	57	32
Ecosoft	65	33
Entelekon	98	48
FairCom	125	51
Gimpel Software	86	—
Gold Hill Computers	Cover III	97
Greenleaf Software Inc.	84	40
Guidelines Software	19	11
Harvard Softworks	44	26
Haventree Software Ltd.	12	57
Impulse Engineering	143	95
IMSI	14	43
Intellicorp	23,24,25	15
JYACC Inc.	108	55
JYACC Inc.	109	56
Kurtzberg Computer Systems	124	50
Lahey	30	18
Language Processors Inc.	131	8
Lattice	95	38
Lifeboat Associates	48	28
Logic Path	40	23
Logicware	87	54
Lugaru Software Ltd.	89	59
Mansfield Software	141	71
Manx Software Systems	13	7
Mark Williams Co.	Cover IV	98
Metagraphics	42	24
MetaWare Incorporated	139	83
Microcompatibles	117	78
Microproducts	62	30
Microsoft	33	—
Microsoft	82,83	39
MicroWay	104	46
Mortice Kern Systems Inc.	122	72
N-TECH	143	91
Nanosoft	43	25
Next Generation Systems	102	52
Norcom	120	64
Northwest Instrument Systems Inc.	16	10
Oakland Group	111	60
Oasys	32	20
Opt-Tech Data Processing	143	92
Pathfinders Software	143	93
Plum Hall	133	80
Polytron	10	5
Production Systems Technologies	79	37
Programmer's Connection	71,72,73	34
Programmer's Paradise	46,47	27
Programmer's Shop	136	82
Programmer's Shop	90,91	44,45
Programs In Motion	15	9
Q-N-E International	38	21
QCAD Systems	144	74
Qualstar Corporation	21	12
Quilt Computing	117	85
Raima Corp.	92	—
Rational Systems	88	42
Realia	121	65
SAS Institute	45	—
Scantel Systems Ltd.	129	73
Seidl Computer Engineering	39	22
Siltronix	139	86
SoftCraft Inc.	3	2
Softfocus	123	67
Software Connection Inc.	128	70
Software Garden	53	29
Softway Inc.	126	62
Solution Systems	81	53
Solution Systems	127	58
Springer Verlag	129	75
Structured Information Systems	143	94
Sutrasoft	97	47
Texas Instruments	28	—
Texas Instruments	6,7	—
Visible Systems Corp.	115	61
Wallsoft Systems Inc.	31	19
Wendin Inc.	29	17
Whitesmiths	113	69
Whitewater Group	26	16

The index on this page is provided as a service to our readers. The publisher does not assume any liability for errors or omissions.



WOOF to ARF ???
Another translation by QPARSER+

Dogs don't use trees for translation, but with QPARSER+, you can use trees to develop compilers, interpreters, complex user-interfaces, language and file format translators (i.e. Pascal to C, Bit Map to Postscript), language debuggers like lint, etc..

QPARSER+ -- don't develop a language-intensive program without it !!

- Companion college text book on compiler design available.
- A new user manual, automated syntax tree construction and an advanced code generation language are just a few of the improvements over our first Translator Writer called QPARSER.
- Widely used in both industrial and educational settings.
- Sample translators included: Pascal, Assembler, etc.
- Variable level debugging and extensive error recovery.
- Full LALR(1) parsing strategy used.
- Runs on 512K PC/XT/AT or VAX/VMS
- Generate full translators in C, Pascal, or any other language.

QPARSER+ -- just $475 (PC/XT/AT)
Call or write for details and a FREE demo disk

QCAD Systems
3333A Octavius Dr, Santa Clara, CA 95054
(408) 727-6671-- outside of California call (800) 538-9787
dBase III © Ahston-Tate; Word © Microsoft; Wordstar © Micropro

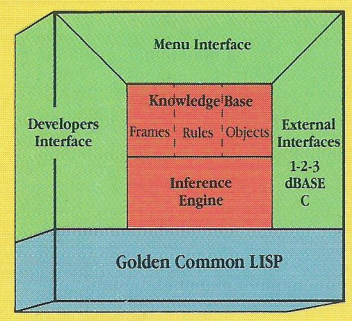
CIRCLE 74 ON READER SERVICE CARD

Gold Hill delivers GoldWorks.™

Now you can build and deliver serious expert systems on advanced PCs.

GoldWorks changes the economics of expert system building.

GoldWorks, formerly code-named Acorn, is designed for professional software developers who need to build serious expert systems and integrate them with conventional applications running on 286- and 386-based PCs. It combines the best features of high-end expert system tools with the ability to develop and deliver expert systems *inexpensively,* on advanced PCs.



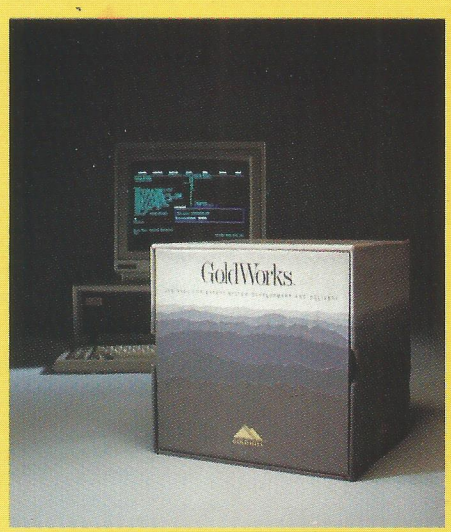
GoldWorks is the most comprehensive expert system tool available for advanced PCs.

Works like a shell.

With the easy-to-use menu interface, you can prototype and build expert system applications quickly, without knowing the underlying programming environment. The **GoldWorks** tutorial, San Marco LISP Explorer® tutorial, on-line help system and example applications help you get started quickly.

Works like a toolkit.

You can access the underlying programming environment to extend and customize the system for your specific applications. And address up to 15 MB of extended memory on the PC AT (more on 386-based PCs).



Works like expert system tools available on high-end workstations . . . at a fraction of the cost.

You get frames with multiple inheritance for flexible knowledge representation. Rules supporting integrated forward and backward chaining for powerful inferencing. Object programming to develop modular applications. Plus advanced features to control the inferencing process. All on the PC you already use.

Works to develop *and* deliver your expert systems.

GoldWorks is the only tool that lets you build and deliver serious expert systems on your PC AT, COMPAQ DESKPRO 386 or Gold Hill's 386 LISP System.

Works to integrate expert systems with conventional PC applications.

You can integrate expert system applications with dBASE III and Lotus 1-2-3 . . . integrate C routines and libraries . . . build and deliver expert systems in network environments.

Works the way you want an expert system builder to work.

GoldWorks sets the standard by which all expert system tools will be compared. Extensively field-tested by developers in dozens of major corporations, **GoldWorks** is backed by Gold Hill's comprehensive customer support, training and consulting programs.

Now, see the functionality and power of GoldWorks for yourself!

Order our unique Demonstration Kit, including full color video preview and complete User's Guide. It's only $49 postpaid, applicable as a credit toward your **GoldWorks** purchase. To order, call toll-free:

1-800-242-5477.

In Mass., call (617) 492-2071.

GoldWorks from Gold Hill. The expert in AI on PCs.

Gold Hill Computers, Inc.
163 Harvard Street
Cambridge, MA 02139

GOLD HILL™

CIRCLE 97 ON READER SERVICE CARD

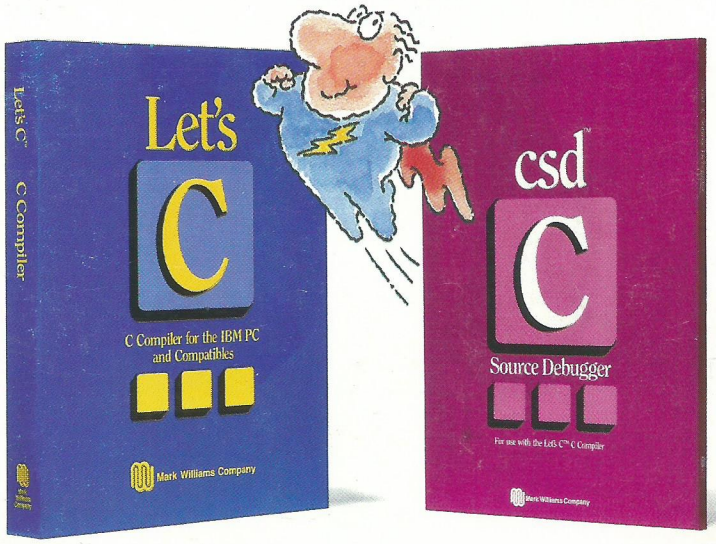
© Copyright 1987, Gold Hill Computers, Inc. Gold Hill, GoldWorks, GCLISP, and 386 LISP System are trademarks of Gold Hill Computers, Inc. San Marco LISP Explorer is a registered trademark of San Marco Associates. Lotus is a registered trademark and 1-2-3 is a trademark of Lotus Development Corporation. dBASE is a trademark of Ashton-Tate. IBM and IBM PC-AT are registered trademarks of International Business Machines Corporation. Intel is a registered trademark and 80386 is a trademark of Intel Corporation. COMPAQ and DESKPRO 386 are trademarks of COMPAQ Computer Corporation.

PRESENTING THE DIFFERENCE BETWEEN FAST COMPILING AND FAST PROGRAMMING.

For compiling speed, you can't do better than Let's C. But to really speed up programming you can't do without the powerful source level debugger, *csd.*

If you want the power, portability and flexibility of C, start with the complete compiler, Let's C. For utilities, editor, compiling speed and fast, dense code, Let's C has it all.

But to get your programs up and running you need more. Because even the fastest compiler can't outrun bugs. You need the revolutionary C Source Debugger, *csd.*



CUT DEVELOPMENT TIME IN HALF WITH *csd*

csd lets you bypass the time consuming frustrations of debugging–like long dumps and clunky assembler. With *csd,* you actually debug in C. You learn faster because you watch your program run in C. You finish faster because *csd* combines the speed of a compiler with the interactive advantages of an interpreter. The end result? Development time is sliced in half.

LIMITED TIME OFFER FREE *csd* WITH LET'S C!

LET'S C AND *csd* FEATURES

NEW VERSION 4.0!

Let's C:

- Now compiles twice as fast
- Integrated edit-compile cycle: editor automatically points to errors
- Includes both small and large memory model
- Integrated environment or command line interface
- 8087 sensing and support
- Documentation features new lexicon format
- MS-DOS object compatible
- New make utility
- Fast compact code plus register variables
- Full Kernighan & Ritchie C and extensions
- Full UNIX compatibility and complete libraries
- Many powerful utilities including make, assembler, archiver, cc one-step compiling, egrep, pr, tail, wc
- MicroEMACS full screen editor with source included
- Supported by dozens of third party libraries
- For the IBM-PC and Compatibles
- Not copy protected

Sieve Benchmark

(Compile time in seconds)
Let's C: **2.8** (On 512K 6Mhz IBM-AT)
Turbo C: **3.89** (As advertised)

csd:

- Large and small memory model
- Debug in C source code, not assembler
- Monitor variables while tracing program
- Does not change program speed or size
- Provides separate source, evaluation, program and history windows
- On-line help screens
- Can interactively evaluate any C expression
- Can execute any C function in your program
- Trace back function
- Ability to set trace points
- Not copy protected

REVIEWERS ARE RAVING ABOUT LET'S C AND *csd.*

"Let's C is an inexpensive, high-quality programming package... with all the tools you will need to create applications."
–William G. Wong, *BYTE,* August 1986.

"The performance and documentation of the $75 Let's C compiler rival those of C compilers for the PC currently being sold for $500...highly recommended..."
–Marty Franz, *PC TECH JOURNAL,* August 1986.

"csd *is close to the ideal debugging environment...a definite aid to learning C and an indispensable tool for program development."*
–William G. Wong, *BYTE,* August 1986.

"This is a powerful and sophisticated debugger built on a well-designed, 'serious' compiler."
–Jonathon Sachs, *Micro/Systems Journal,* April, 1986

START TO FINISH, THERE'S NO BETTER ENVIRONMENT.

Get started with the right C compiler and you'll have everything you need for development–including source level debugging. On top of it all, Let's C and *csd* are today's best values in professional C programming tools. And most reliable: Mark Williams C compilers have been sold with DEC, Intel and Wang computers since 1981.

60 DAY MONEY BACK GUARANTEE

Mark Williams gives you a full 60 days to find out just how good Let's C and *csd* really are–or your money back.

So if you want more than a fast compiler–if you want your programs up and running fast, ask for Let's C and *csd.* You'll find them at your software dealer's, in the software department of your favorite bookstore, through the Express Program at over 5500 Radio Shacks or you can order now by calling 1-800-MWC-1700.*

*In Illinois call, 1-312-472-6659.

Mark Williams Company
1430 West Wrightwood, Chicago, Illinois 60614

© 1987 Mark Williams Company
Let's C is a registered trademark of the Mark Williams Company.
UNIX is a trademark of Bell Labs.

MARK WILLIAMS LET'S C AND *csd.* ONLY $75 EACH.

CIRCLE 98 ON READER SERVICE CARD